ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA DE SISTEMAS

STAR OGRE. VIDEOJUEGO RUN N GUN
EN 3D CON LA BIBLIOTECA LIBRE OGRE

Álvaro Verástegui Barchilón

Cádiz, Octubre 2011

ESCUELA SUPERIOR DE INGENIERÍA

INGENIERO TÉCNICO EN INFORMÁTICA

DE SISTEMAS

STAR OGRE. VIDEOJUEGO RUN N GUN EN 3D CON LA BIBLIOTECA LIBRE OGRE.

DEPARTAMENTO: LENGUAJES Y SISTEMAS INFORMÁTICOS

DIRECTORES DEL PROYECTO: MANUEL PALOMO DUARTE
JUAN MANUEL DODERO BEARDO

AUTOR DEL PROYECTO: ALVARO VERÁSTEGUI BARCHILON

Cádiz, Octubre 2011

Fdo: Álvaro Verástegui Barchilón

A mi tutor, Manuel, por iniciarme en el camino.

A mi Madre, por ayudar a permitirme andarlo.

A Belén, por darme el empujón necesario para recorrerlo.

A todos, Gracias por aguantarme este tiempo.

Licencia

Índice general

Índice de Figuras y Tablas

Capítulo 1

Introducción

Este proyecto trata sobre el desarrollo de un videojuego de acción utilizando el motor Ogre3D. Este motor es software libre, y cuenta con una comunidad abierta en internet muy amplia. Durante el desarrollo del proyecto se aprovechará la experiencia del autor en el mundo del desarrollo profesional de videojuegos para indicar las partes más importantes en el desarrollo, así como los perfiles que contribuyen a la realización de un software tan complejo como es un videojuego. Por ello es que en determinados temas pueden no incluirse referencias, al ser experiencia propia.

1.1 Objetivo del proyecto

El objetivo principal es crear un juego de naves en tercera persona, donde el jugador pase un buen rato enfrente de la pantalla del ordenador matando el mayor número de naves posible. Se intentará hacerlo divertido, con mucha acción, con el objetivo de mostrar paso a paso, el desarrollo de un videojuego, desde su concepción hasta su entrega final.

El jugador podrá jugar el modo historia, donde luchará con la nave a lo largo de dos escenarios, tratando de llegar al final sin morir; o jugar el modo arcade, en un escenario infinito, donde cada vez salen más naves

con mayor nivel de dificultad, hasta que termine la última vida, tratando de hacer el mayor número posible de puntos.

1.2 Estructura de la memoria de proyecto

A lo largo de éste capítulo, se ha dado una introducción acerca del contexto en el que nos movemos. En el siguiente capítulo, se explican los conceptos básicos, mostrando los elementos y las tecnologías del desarrollo de un videojuego.

En el capítulo 3 se muestra la planificación que se ha seguido para el completo desarrollo del juego. A lo largo del capítulo 4 se realiza el análisis del proyecto, incluyendo los casos de uso posibles en el programa. En el siguiente capítulo, el 5, se realiza el diseño del proyecto, incluyendo especificaciones, requisitos, interfaces y funcionalidades. En el capítulo 6, se muestran las soluciones que se han tomado para desarrollar la implementación de cada modulo. Después, en el capítulo 7, se muestra un plan de pruebas, para asegurar el correcto funcionamiento del juego. En el capítulo 8 se añade un glosario con las definiciones más importantes a tener en cuenta acerca de un desarrollo de videojuegos.

Por último se incluye un capítulo, el 9, en el que se incluyen las conclusiones finales del proyecto, siguiendo con el capítulo 10, en el que se anexan documentos importantes del juego como son el documento de concepto o el manual de instalación y de usuario.

Capítulo 2

Conceptos básicos

A continuación se explican conceptos básicos en el desarrollo de un videojuego. Hablaremos sobre el ciclo de vida de un proyecto software de un videojuego, las figuras principales que participan en el desarrollo, veremos un análisis de cómo los medios han tratado a los videojuegos, pasaremos a ver las tecnologías utilizadas y para terminar, veremos el principal precedente de este juego, el StarFox64 de la Nintendo64.

2.1 Ciclo de vida de un desarrollo

Como todo desarrollo software, un videojuego pasa por un ciclo de vida. Lo más normal es que sea uno de tipo secuencial-iterativo, sobre todo en la época de prototipado, a lo largo del cual se van construyendo las diferentes mecánicas y los distintos elementos del juego, para aceptar unos y descartar otros.

Las fases del desarrollo, por tanto, son las siguientes:

- Concepto

A lo largo de esta fase, el equipo de diseño se encarga de formular ideas sobre el videojuego que se quiere crear. Aquí lo importante son las ideas, resumir el mayor número de palabras en un concepto propiamente dicho. Se trata de formular una base mínima, compacta,

que contenga la esencia del proyecto. Ejemplo: Super Mario, juego de plataformas para GameBoy/ Nes/ Wii/ PC.

Todas las ideas que no entren en el documento se descartan, porque podrían estropear el desarrollo del juego en el futuro. Aquí el equipo de programación comienza a formarse sobre las tecnologías que se van a utilizar.

- Prototipado

En esta etapa, el equipo de diseño debería tener claro sobre qué idea básica se va a trabajar. Aquí, el grupo de diseño elabora mecánicas básicas que le podrían sentar bien al juego. Si el jugador podrá recoger ítems, usar llaves para abrir puertas, volar, etc... Todas ellas son implementadas por el equipo de programación, con un aspecto visual muy básico, mediante lo que se conoce como "Place holders", que son elementos que visualmente no encajan con el juego para que se sepa que están mal puestos. Esto se hace para que el equipo de diseño vea cuales son más factibles, mediante una jugabilidad básica, de incorporarlas en el juego finalmente.

Cabe destacar que aunque es un proceso donde se prueban hasta las mecánicas que más se ajustan al juego, muchas de ellas, aunque interesantes, se descartan para que el juego sea realmente manejable y divertido para el jugador. A diferencia de la fase de Concepto, estas ideas se guardan, pues si hay posibilidades presupuestarias y temporales podrían incluirse en el juego (o emplearse en una segunda parte o extensión).

En esta fase, el equipo de desarrollo trabaja activamente con Diseño para implementar los ejemplos mientras se sigue formando en tecnología y recursos adicionales, como exportadores de modelos entre sistemas, herramientas auxiliares...

- Diseño

En la fase de Diseño, propiamente dicha, se ponen por escrito ideas aprobadas y aceptadas en prototipado, detallando cada una de ellas. La redacción debe de ser clara y precisa, evitando ambigüedades, puesto que son la base de la comunicación entre el equipo de diseño y el de programación, ya que cuando se quiera consultar que es lo que se quiere en concreto, se consultará el documento correspondiente.

El equipo de programación irá implementando la arquitectura del juego, y se irán implementando todas aquellas herramientas y programas externos a la aplicación, que luego el equipo de arte y de diseño utilizará para completar el juego.

- Producción

En esta fase el juego ha quedado casi cerrado en cuanto al diseño. Únicamente quedarán por pulir aspectos dependientes de la experiencia final del juego en conjunto, pero deberían ser aspectos muy concretos. Cómo por ejemplo, que el jugador pueda estar debajo del agua 10 o 20 segundos. Lo que no debería diseñarse en esta fase nunca es, si estamos hablando de un juego de simulación de carreras tipo Gran turismo, muy realista, convertirlo en uno de carreras tipo Need For Speed, con mucha acción y menos realismo. Porque corremos el riesgo en demorarnos en exceso con el desarrollo.

De hecho, uno de los grandes problemas que sufre el desarrollo de los videojuegos hoy en día, suelen ser los cambios de diseño a última hora. Esto provoca juegos producidos fuera de plazo, con presupuesto superado, o con deficiencias en su funcionalidad (que no ha sido probada adecuadamente).

- Pre-Alpha, Alpha

Estas dos fases suelen ser una sola, pero hay casos en los que se realiza una pequeña iteración primero. Al final de esta fase se han terminado de programar todas las funcionalidades. Todas las misiones,

niveles y scripts están completos y funcionando en todos los niveles de dificultad. El código está razonablemente optimizado y estable tras una fase de prueba informal.

Es deseable que al final de esta fase el juego se pueda instalar mediante un asistente/instalador.

- Pre-Beta, Beta

Lo mismo que en la anterior, suelen ser la misma fase. También se suele llamar content-complete, por lo que significa. Se van preparando todos los procesos de masterización (o masterización) y se termina de rellenar el contenido del juego: textos, localización, sonidos...

Al final de esta fase tenemos el juego completo en su soporte final (CD/DVD, cartucho, etc), instalable y jugable en todas las fases y dificultades.

- Releases Candidates

Una release candidate es una versión que es candidata a ser la definitiva. Suele haber muchas de ellas (quizás centenares), puesto que el equipo de prueba (Quality Assurance, QA) testea el juego minuciosamente. Se recorre cada rincón del mapa, se salta en cada colina de varias formas, se comprueba el comportamiento de cada colisión... Con la información que se va recabando se reporta hasta el menor fallo. Hay que aclarar que no es necesariamente malo que se encuentren fallos en un producto, puesto que los fallos los deben encontrar los testers, no los usuarios finales. Este proceso suele ser el más tedioso para el equipo de programación y debe hacerse con tacto, porque a nadie le gusta que le muestren sus fallos (aunque se admitan).

Al final de esta fase el juego está tal y como debería llegar finalmente al usuario final.

- Gold Master

En esta fase por fin, el juego queda libre de errores y se envía a duplicación, que es la fase que se encarga de realizar las copias del juego. En el caso de consola, a veces las condiciones de uso de los entornos de desarrollo obligan a que el fabricante tenga que dar el visto bueno ("certificación") al juego, indicando que cumple con los estándares de la licencia.

- Mantenimiento

En esta fase se hace un seguimiento de la experiencia de los usuarios al usar el juego. Incluye parches para arreglar errores que no se detectaron en las pruebas internas, o en ocasiones se implementan mejoras, webs con los mejores resultados, etc...

- Post-Mortem

Esta fase consiste en indicar por parte de todo el equipo, que elementos del desarrollo fueron bien, y cuáles fueron mal. Este documento es realmente útil de cara a futuros proyectos, puesto que poniendo por escrito aquellos elementos que destacan, tanto para bien como para mal, se potencian los positivos y se intenta mejorar en los negativos.

Este documento se realiza idealmente entre dos semanas y dos meses después de cerrar el proyecto. Resulta de gran valía de cara a futuros proyectos para intentar no cometer los mismos errores, y aprovechar las ventajas de lo que salió bien. Aunque hay que tener precaución, puesto que lo que salió bien en un proyecto, puede no resultar beneficioso para el siguiente.

2.2 Perfiles básicos del desarrollo

A continuación se describen los perfiles básicos que intervienen en un desarrollo de videojuego[1].

- Programador

Es el que se encarga de realizar todo el código asociado a la aplicación. Esto incluye herramientas, infraestructura y por supuesto el código del juego. Básicamente, su misión consiste en implementar las ideas del diseñador, mediante la ayuda de los gráficos que proporciona el grafista. Solían ser gente autodidacta que aprendía a programar por su cuenta. Hoy en día suelen ser ingenieros informáticos con alguna especialidad concreta.

- Grafista

Básicamente dibuja y/ o modela todos los elementos del juego. Escenarios, personajes, objetos… Se encargan de la parte 2D y 3D del juego. Por encima del grafista suele haber uno o varios directores de arte que son los que dirigen las líneas, junto a los diseñadores, de cómo es el arte del juego. Este perfil lo suele tener gente de bellas artes, aunque cada vez menos.

- Diseñador

Su misión consiste en idear el juego. Desde el tipo de juego, pasando por cuantos personajes, sus características, la vida, los elementos informativos fijos de la interfaz (Head-Up Display, HUD), la ambientación… Básicamente son los que "crean" el juego. Junto a los artistas definen cuales son los elementos que se pintan en pantalla. También definen los comportamientos de los personajes no jugadores, bien mediante scripts o bien mediante patrones de comportamiento.

[1] Estos perfiles son fruto de la experiencia del autor.

- Productor

Persona encargada de gestionar los equipos, así como de hacer que se cumplan los plazos. Son los que se encargan de lidiar con los editores, así como de gestionar los recursos humanos, ver si hace falta más gente para incorporar al equipo y asignar los plazos de entrega.

2.3 Clasificación de un videojuego

Vamos a establecer una pequeña clasificación de los videojuegos en función a dos criterios, por una parte el tipo de juego que es y por otra el tipo de género.

2.3.1 Tipos de juego

- Redes Sociales

Con el auge hoy en día de las redes sociales (Facebook, Twitter, Tuenti…), son muchas las empresas, grandes y no tan grandes, que han visto la oportunidad de modelo de negocio en ellas. Gente que habitualmente no jugaba a los videojuegos, o bien gente que sí jugaba, pero ha crecido y ya no dispone de tanto tiempo, son sus principales consumidores. Ofrecen partidas rápidas y limitadas, en las que el usuario puede avanzar en el juego mediante una serie de acciones sencillas, que no requieren demasiado esfuerzo por parte del jugador. Cómo ejemplo de grandes empresas tenemos a Zynga, actualmente la mayor empresa de juegos sociales, o no tan grandes como Digital Chocolate, cuyos juegos tienen una cantidad considerable de usuarios diarios.

- Portátiles

Son juegos, que sin llegar a la simplicidad de los juegos de redes sociales, si que están concebidos para ser jugados en ocasiones donde el tiempo no es tan amplio como se quisiera. Viajes de autobús, viajes en tren, algún rato suelto en casa. Son las situaciones en las que se suele

requerir a este tipo de juegos. Dentro de esta categoría tendríamos los juegos para móviles, para tablets, o bien para las portátiles de Sony y Nintendo en la actualidad.

- Sistema Fijo

Son aquellos que se juegan en una consola de sobremesa o en un ordenador. Básicamente son aquellos juegos que requieren de un método de entrada mediante un mando, un teclado, un ratón o una cámara, conectados a una consola o el ordenador. Este tipo de juegos, requieren una cantidad de tiempo mayor que los dos anteriores, por lo que el público objetivo se reduce bastante.

2.3.2 Géneros

- Rol

Un videojuego de Rol, o más conocido por las siglas RPG que significan *Role Player Game,* es un videojuego donde el jugador asume el papel de uno o más personajes, y desarrolla sus habilidades. Cabe destacar, que en este tipo de juegos la carga de personalidad que tienen los personajes es más intensa que podría ser en el resto de juegos. Suelen tener una gran duración, en torno a las 35-40 horas.

- Lucha

Son aquellos que enfrentan al jugador contra uno o muchos enemigos. Básicamente tenemos dos tipos de juegos, uno contra uno, o uno contra todos. Normalmente los primeros se desarrollan por rondas, encima de un ring. Los segundos suelen ser en un escenario abierto, o no, en los cuales hay que llegar de un punto a otro del escenario matando a todos los enemigos. La duración de estos juegos es variable, aunque no suele llegar a la duración de un juego de rol.

- Simulación

Este tipo de juego es aquel que presenta un escenario donde lo importante es el realismo. Los hay de todo tipo de ambientaciones,

desde simuladores deportivos, de aviación, de carreras de coches o hasta de submarinismo o de pesca.

- Deportes

Son un subtipo de simulación, pero que merecerían un tipo aparte ya que son muy numerosos. Son aquellos juegos que están basados en un deporte real o inventado, siguiendo sus normas o modificándolas. Juegos de fútbol, baloncesto o tenis, son ejemplos de este tipo.

- Plataformas

La finalidad de este tipo de juegos es avanzar de un punto a otro del escenario, saltando y superando obstáculos. Normalmente el objetivo es conseguir el mayor número de puntos, algún objeto, resolver algún puzzle...

- Educativos (serious games)

Juegos donde la finalidad es conseguir que el jugador aprenda algo. Bien sea algún idioma, cocinar, hábitos de salud, matemáticas... Están orientados a todo tipo de usuarios y lo normal es que suelan ser divertidos, para que la tarea cognitiva sea más efectiva.

- Puzzle

Este tipo de juegos suele presentar al jugador una serie de enigmas y retos a resolver, bien sea mediante preguntas directas, o bien mediante acertijos escondidos en algún lugar del escenario. La temática de este tipo de juegos es de lo más variada. Desde juegos musicales, de abogados, de medicina, de historia... Cualquier ambientación es válida.

- Estrategia

En los juegos que son de tipo estrategia, el jugador suele representar a una figura dominadora, en la que controla a una serie de unidades

para lograr un objetivo determinado. Desde ganar una batalla, hasta jugar un partido de fútbol o una partida de cartas.

- Aventuras gráficas

Este es un género que vivió su momento álgido a principios de los 90, cuando el PC era el principal medio de ocio digital. Consiste en jugar una historia narrada, mediante una serie de acciones predefinidas que el jugador va escogiendo, obteniendo diferentes resultados. Normalmente el jugador debe recopilar una serie de objetos usándolos en determinados momentos, para llegar al final de la historia. En este género, merece mención la empresa LucasArts, que supo crear clásicos dentro de este género donde, de entre todos, destacan la saga Monkey Island o la saga de Indiana Jones.

2.4 El videojuego en los medios[2]

Hoy en día el videojuego goza de una repercusión muy buena en los medios, tanto tradicionales (televisión, radio, prensa…), como especializados (blogs, páginas de internet, revistas de videojuegos…). Pero esto no ha sido así siempre, en realidad no hace mucho tiempo que disponemos de esta situación. Fuera de España, el videojuego estaba bastante aceptado socialmente, sobre todo en países como Japón o Estados Unidos, donde han gozado de una industria desde hace bastante tiempo.

Aquí, los videojuegos era una cosa más de estar en salones recreativos, que de estar en casa encerrado. Con el tiempo, y la aparición de consolas más evolucionadas, la cosa fue cambiando. Poco a poco fueron llegando las primeras consolas, fueron apareciendo las primeras revistas de videojuegos, que como los mismos editores

[2] Este apartado se basa en la experiencia y las vivencias del autor, además de diversas lecturas en varios medios. No hay una referencia clara.

cuentan, el resto de la comunidad editorial les miraba por encima del hombro.

Decían frases como “Si solo se dedican a jugar todo el día, eso no es un trabajo”, sin apenas fijarse en lo que hacían, o el trabajo que suponía tener que investigar sobre una cosa en la que apenas nadie escribía.

A principios de los años 90, comienza a haber en España un movimiento editorial consolidado. Los niños empiezan a jugar cada vez a edades más tempranas, y los videojuegos se cuelan en los patios del colegio.

A finales de los años 90, Sony, que andaba trabajando codo con codo con Nintendo para sacar al mercado una consola que utilizara su tecnología de los CD-Roms, finalmente, por decisiones empresariales, lanza en solitario la PlayStation y esto supone una irrupción total en los hogares. Pero era un fenómeno tan grande, que apenas la gente conocía bien. Y es por eso, que los medios tradicionales intentan formar parte del fenómeno, pero a su manera. Fue habitual ver reportajes y artículos hablando del nuevo fenómeno, y demonizando aquello que se escapaba a su conocimiento.

Con la llegada de internet, y la globalización de la tecnología, el mundo de los videojuegos se ha conocido en mayor medida, también gracias en parte a que los chavales que, de pequeños jugaron a los videojuegos, hoy han crecido y ya tienen conciencia de este mundo, de manera que los videojuegos, en la actualidad, gozan de una reputación muy buena.

2.5 Tecnologías utilizadas

A continuación se describen las tecnologías y los programas utilizados más relevantes.

- OGRE 3D

[WikiEs] **OGRE 3D** (acrónimo del inglés *Object-Oriented Graphics Rendering Engine*) es un motor de renderizado 3D orientado a escenas, escrito en el lenguaje de programación C++.

Sus bibliotecas evitan la dificultad de la utilización de capas inferiores de librerías gráficas como OpenGL y Direct3D, y además, proveen una interfaz basada en objetos del mundo y otras clases de alto nivel.

El motor es software libre, licenciado bajo MIT y con una comunidad muy activa. Incluso ha sido utilizado en algunos videojuegos comerciales, como por ejemplo *Torchlight*.

Como su nombre indica, OGRE sólo es un motor de renderizado, esto es, un motor que sólo se encarga de pintar las cosas en escena, no proporciona ninguna herramienta adicional para la simulación, las físicas o el audio, por ejemplo

- PhysX

Para solucionar el tema de la física, se ha recurrido a un motor de física propietario de Nvidia. [WikiEs] **PhysX** es un motor propietario de capa de software intermedia o "middleware" y un kit de desarrollo diseñados para llevar a cabo cálculos físicos muy complejos.

Los motores físicos de middleware permiten a los desarrolladores de videojuegos abstraerse durante el desarrollo, ya que PhysX proporciona funciones especializadas en simulaciones físicas complejas, lo cual da como resultado una alta productividad en la escritura de código.

- Fmod

[WikiEs] **FMOD** es una librería de audio propietaria hecha por Firelight Technologies que reproduce música de diversos formatos en muchos sistemas operativos, que se usa en juegos y software para proporcionar sonido.

FMOD tiene una *Non-Commercial License*, licencia no comercial, para permitir utilizar el software gratis a proyectos que no tienen una distribución comercial.

- 3ds Max

[WikiEs] **Autodesk 3ds Max** es uno de los programas de animación 3D más utilizados. Dispone de una sólida capacidad de edición, una omnipresente arquitectura de plugins y una larga tradición en plataformas Microsoft Windows. 3ds Max es utilizado en mayor medida por los desarrolladores de videojuegos, aunque también en el desarrollo de proyectos de animación como películas o anuncios de televisión, efectos especiales y en arquitectura.

Este programa es uno de los más reconocidos modeladores de 3d masivo, habitualmente orientado al desarrollo de videojuegos, con el que se han hecho enteramente títulos como las sagas 'Tomb Raider', 'Splinter Cell' y una larga lista de títulos de la empresa Ubisoft.

2.6 Antecedentes

El principal juego que tiene de referencia es el StarFox 64 (Nintendo EAD, 1997). Juego de naves en tercera persona en el que encarnamos a un mercenario espacial, al que le encargan diversas misiones para salvar diversos planetas. Según el jugador va pasando niveles y, en función de lo bien que lo haga, podrá ir por un camino u otro.

Este juego supuso una revolución en el mundo de los videojuegos gracias, en parte, a que fue de los primeros de la consola Nintendo 64, y explotaba un universo en 3D muy bien logrado.

Capítulo 3

Planificación

A continuación se detalla la planificación del proyecto modularmente, y se incluye un diagrama de Gantt que ilustra mejor el proceso. El juego ha sido desarrollado en 11 meses y medio en las fases habituales en este tipo de proyectos, concepto, diseño, producción y release. El periodo de concepto vino precedido por un periodo de investigación sobre las tecnologías principales a utilizar.

3.1 Planificación temporal

A continuación se describe la planificación del proyecto. Pero antes hay que hacer una aclaración. Esta planificación que se realiza es sobre el juego en concreto pero, antes de comenzar, el juego paso por varias cancelaciones antes de comenzar a desarrollarse. Ese tiempo no está incluido en la planificación, puesto que no forma parte del desarrollo del juego.

- **Pre-Concepto**: 35 días

Durante esta fase, se investigó sobre las tecnologías que se iban a utilizar, en concreto aquellas sobre las que menos se conocía, por lo menos no lo suficiente para realizar un proyecto de este tipo, OGRE y 3ds Max. En concreto, del primero se consultó la página web oficial y se realizaron los tutoriales que vienen en la página. Para investigar 3ds

Max, se usó el programa y se visualizaron videotutoriales sobre animaciones y texturización.

- Aprendizaje de OGRE: 22 días.
- Aprendizaje de 3ds Max: 13 días.

- **Concepto**: 3 días.

En esta fase se establecen las líneas generales del juego que se va a realizar. Normalmente en el documento de concepto se establecen las líneas generales de las mecánicas, y se establecen esbozos de lo que va a ser el juego. El resultado de esta fase está incluido en el Anexo 1 de este documento.

- Diseño del concepto: 1 día.
- Redacción del documento: 2 días.

- **Diseño**: 30 días.

Durante esta fase se desarrolla el documento de concepto, extendiendo que se entiende de cada mecánica y estableciendo un plan de ataque a los requisitos. Se suele realizar una planificación, estableciendo cuanto tiempo se estima que va a durar el desarrollo. Siendo un juego de naves, que desarrolla sus mecánicas en el espacio, principalmente se han simplificado bastante los requerimientos como, por ejemplo, la inteligencia artificial de los enemigos. Con esto no se quiere decir que la inteligencia artificial sea poco inteligente, sino que es menos avanzada que cómo podría ser en un juego de tipo espionaje, por poner un ejemplo.

- Arte: 5 días.
    - Personaje principal: 1 día.
    - Enemigos: 1 día.
    - Escenarios: 1 día.
    - Ambientación: 2 días.

- Mecánicas: 8 días.
    - Disparo: 1 día.
    - Freno/Aceleración: 1 día.
    - Movimiento: 2 días.
    - Items: 2 días.
    - Modos de juego: 2 días.
- Escena: 6 días.
    - Elementos en pantalla: 4 días.
    - Dimensión del escenario: 2 días.
- Interfaz del Usuario: 11 días.
    - Interfaz de pantalla (HUD): 9 días.
    - Controles: 2 días.

- **Producción**: 230 días.

La fase de producción es la fase que más tiempo ocupa, ya que es la fase en la que el juego toma forma y donde realmente se observan los frutos del diseño. Si bien es cierto que un juego mal diseñado es malo, un juego mal producido no es ni juego. También hay que añadir que los fallos de programación se ven en seguida, mientras que los fallos de diseño se ven más a la larga.

Una vez se tiene claro que juego se va a hacer, se comienza a desarrollar teniendo en cuenta que el diseño no es intratable ni inmodificable, pero tampoco todo lo contrario, es decir, modificable al 100%. Hay que programar teniendo en cuenta posibles alteraciones que se pueden producir, intentando que no se produzcan a última hora, estar abierto a cambios y, sobre todo, pensar que los cambios se hacen pensando en ir a mejor siempre.

- Física: 18 días.
    - Acoplamiento de PhysX al proyecto: 7 días.
    - Implementación del gestor de colisiones: 5 días.
    - Implementación del módulo de físicas: 6 días.
- Aplicación: 23 días.
    - Creación de la máquina de estados: 6 días.
    - Estado Menú: 7 días.
    - Estado Juego: 10 días.
- Lógica: 40 días.
    - Entidades: 2 días.
    - Enemigo: 6 días.
    - Jugador: 6 días.
    - Servidor de recursos estáticos: 8 días.
    - Gestor de niveles: 9 días.
    - Partida: 9 días.
- Gráficos: 119 días.
    - Gestor de gráficos: 5 días.
    - Entidad gráfica: 2 días.
    - Entidad nave: 3 días.
    - Entidad ítem: 2 días.
    - Constantes: 1 día.
    - Creación de recursos gráficos: 106 días.
- Inteligencia Artificial: 20 días.
    - Investigación: 4 días.
    - Árboles de comportamiento: 16 días.
- Búsqueda de recursos: 10 días.
    - Sonidos: 4 días.
    - Gráficos para texturas: 6 días.

- **Alfa, Beta y Gold**: 16 días.

La penúltima fase del proyecto es la que idealmente debe durar menos, junto a la de concepto. Si esta fase dura demasiado, es que durante la fase de producción no se realizó bien. Bien porque el equipo de QA no probó bien las cosas, o bien porque el equipo de programación optimizó mal el juego, por falta de comunicación con diseño o por el motivo que sea. Los anteriores son sólo unos pocos ejemplos de los problemas que suelen alargar esta fase.

Para lograr el objetivo de que esta fase sea realmente corta y cumplir con los plazos establecidos es importante que se cumplan tres cosas:

a) El equipo de diseño tenga buena comunicación con el resto, sea proactivo y tenga las ideas lo más claro posible antes de llevar ninguna idea a cabo.
b) El equipo de programación sepa programar de una manera relativamente óptima, cumpliendo los objetivos marcados y siendo abierto a cambios factibles en el diseño.
c) El equipo de QA pruebe el desarrollo completo de la manera más minuciosa posible, pero entendiendo correctamente qué es un fallo y qué es un comportamiento razonable.

Cumpliendo esos tres requisitos de una manera razonable, es más que probable que el juego se termine en tiempo, coste y con los objetivos marcados a corto plazo. Obviamente hay circunstancias imprevistas que impiden el desarrollo normal de cualquier proyecto, pero eso se puede prevenir realizando un buen análisis de riesgos, incluido en el documento de concepto e incidiendo sobre los posibles efectos negativos desde el principio del desarrollo.

- Pruebas: 10 días.
- Manuales: 4 días.
  - Instalación: 2 días.
  - Usuario: 2 días.
- Release: 2 día.

- **Memoria del proyecto**: 25 días.

Por último, la redacción de la memoria del proyecto, donde se detallan todos los aspectos importantes del desarrollo y se incluye un post-mortem al final de ésta, en el Anexo 4.

Tiempo total del proyecto: 339 días.

3.2 Diagrama de Gantt

A continuación se muestra un diagrama de Gantt exponiendo la planificación que se ha descrito anteriormente. Primero de forma completa y después detallado por partes.

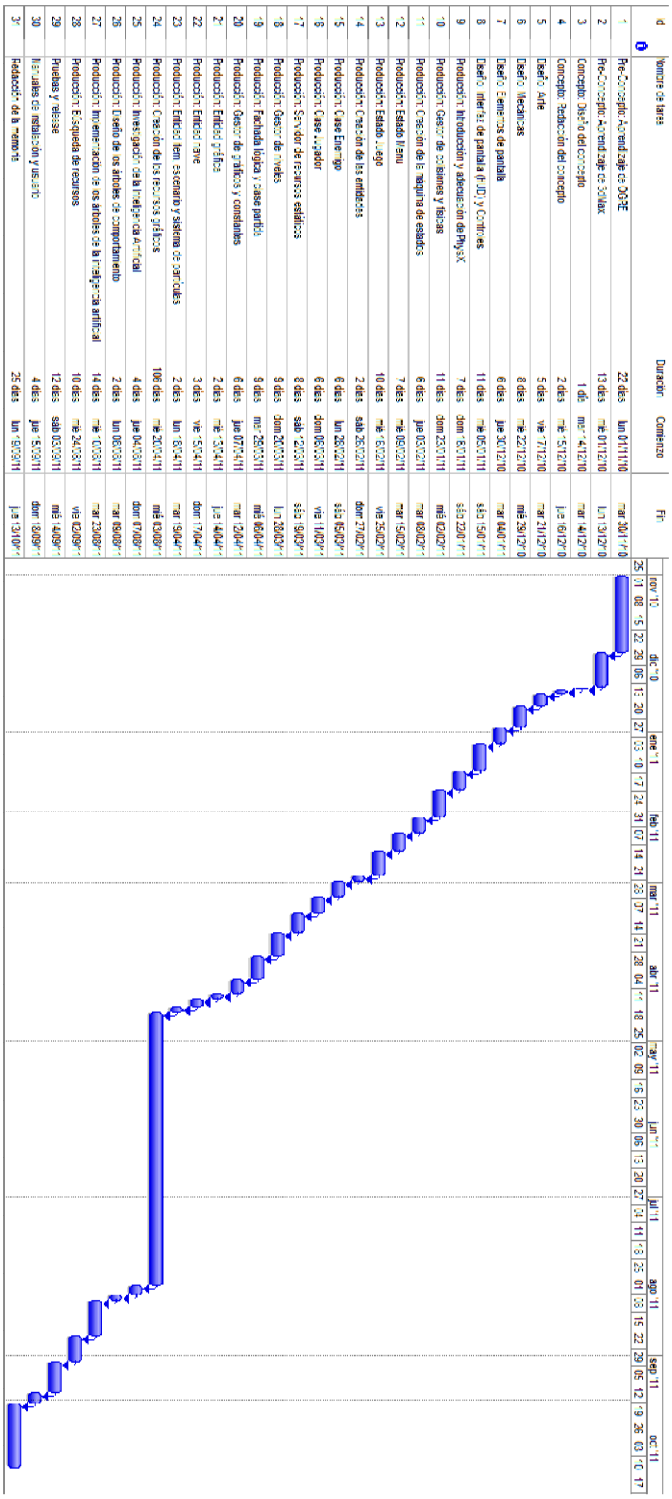

Figura 3.1. Diagrama de Gantt completo

Id	Nombre de tarea	Duración	Comienzo	Fin
1	Pre-Concepto: Aprendizaje de OGRE	22 días	lun 01/11/10	mar 30/11/10
2	Pre-Concepto: Aprendizaje de 3dMax	13 días	mié 01/12/10	lun 13/12/10
3	Concepto: Diseño del concepto	1 día	mar 14/12/10	mar 14/12/10
4	Concepto: Redacción del concepto	2 días	mié 15/12/10	jue 16/12/10
5	Diseño: Arte	5 días	vie 17/12/10	mar 21/12/10
6	Diseño: Mecánicas	8 días	mié 22/12/10	mié 29/12/10
7	Diseño: Elementos de pantalla	6 días	jue 30/12/10	mar 04/01/11
8	Diseño: Interfaz de pantalla (HUD) y Controles	11 días	mié 05/01/11	sáb 15/01/11

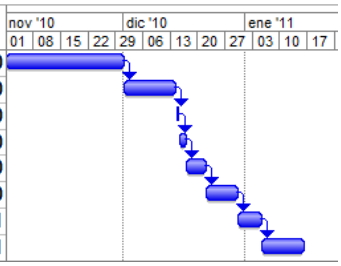

Figura 3.2. Diagrama de Gantt – Parte 1.

Id	Nombre de tarea	Duración	Comienzo	Fin
9	Producción: Introducción y adecuación de PhysX	7 días	dom 16/01/11	sáb 22/01/11
10	Producción: Gestor de colisiones y físicas	11 días	dom 23/01/11	mié 02/02/11
11	Producción: Creación de la máquina de estados	6 días	jue 03/02/11	mar 08/02/11
12	Producción: Estado Menu	7 días	mié 09/02/11	mar 15/02/11
13	Producción: Estado Juego	10 días	mié 16/02/11	vie 25/02/11
14	Producción: Creación de las entidades	2 días	sáb 26/02/11	dom 27/02/11
15	Producción: Clase Enemigo	6 días	lun 28/02/11	sáb 05/03/11
16	Producción: Clase Jugador	6 días	dom 06/03/11	vie 11/03/11

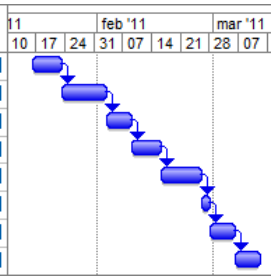

Figura 3.3. Diagrama de Gantt – Parte 2.

Id	Nombre de tarea	Duración	Comienzo	Fin
17	Producción: Servidor de recursos estáticos	8 días	sáb 12/03/11	sáb 19/03/11
18	Producción: Gestor de niveles	9 días	dom 20/03/11	lun 28/03/11
19	Producción: Fachada lógica y clase partida	9 días	mar 29/03/11	mié 06/04/11
20	Producción: Gestor de gráficos y constantes	6 días	jue 07/04/11	mar 12/04/11
21	Producción: Entidad gráfica	2 días	mié 13/04/11	jue 14/04/11
22	Producción: Entidad nave	3 días	vie 15/04/11	dom 17/04/11
23	Producción: Entidad item, escenario y sistema de partículas	2 días	lun 18/04/11	mar 19/04/11

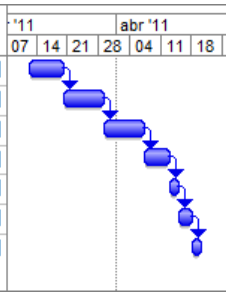

Figura 3.4. Diagrama de Gantt – Parte 3.

Id	Nombre de tarea	Duración	Comienzo	Fin
24	Producción: Creación de los recursos gráficos	106 días	mié 20/04/11	mié 03/08/11

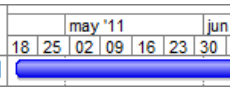

Figura 3.5. Diagrama de Gantt – Parte 4.

Id	Nombre de tarea	Duración	Comienzo	Fin
24	Producción: Creación de los recursos gráficos	106 días	mié 20/04/11	mié 03/08/11
25	Producción: Investigación de la Inteligencia Artificial	4 días	jue 04/08/11	dom 07/08/11
26	Producción: Diseño de los árboles de comportamiento	2 días	lun 08/08/11	mar 09/08/11
27	Producción: Implementación de los árboles de la inteligencia artificial	14 días	mié 10/08/11	mar 23/08/11
28	Producción: Búsqueda de recursos	10 días	mié 24/08/11	vie 02/09/11
29	Pruebas y release	12 días	sáb 03/09/11	mié 14/09/11
30	Manuales de instalación y usuario	4 días	jue 15/09/11	dom 18/09/11
31	Redacción de la memoria	25 días	lun 19/09/11	jue 13/10/11

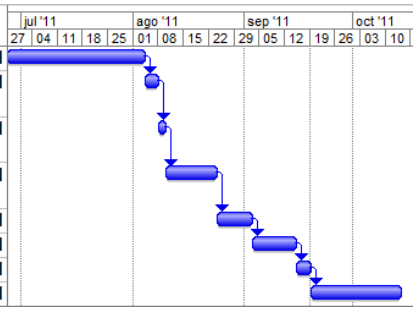

Figura 3.6. Diagrama de Gantt – Parte 5.

Capítulo 4

Análisis

En este capítulo analizaremos la estructura general del juego planteando los posibles casos de uso que el usuario se encontrará y, después, mostraremos mediante el uso de un esquema, cómo están relacionadas las clases del sistema internamente, primero dentro de cada módulo y después como se relacionan los módulos entre sí.

4.1 Casos de uso

A continuación se muestra un diagrama de casos de uso para luego explicar cada uno de ellos.

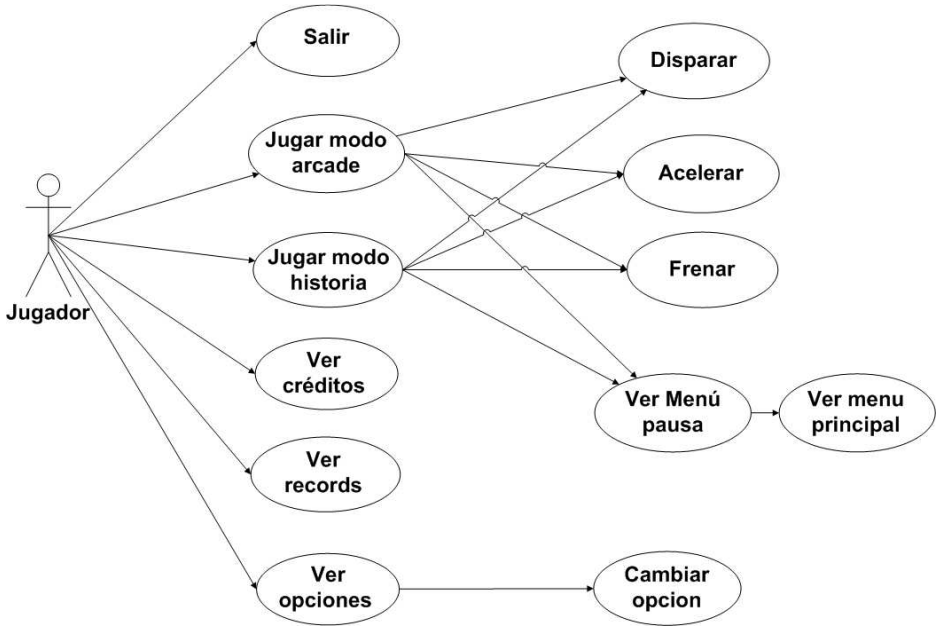

Figura 4.1. Diagrama de casos de uso.

Caso de Uso 1. Comenzar partida en modo historia

Escenario principal

1. Se muestra el menú principal.
2. El jugador selecciona la opción “Jugar”.
3. Se muestra el menú de selección de modo de Juego.
4. El jugador selecciona la opción “Historia”.
5. Se inicia el juego.

Escenario alternativo

1. Se muestra el menú principal.
2. El jugador selecciona la opción “Jugar”.
3. Se muestra el menú de selección de modo de Juego.
4. El jugador selecciona la opción “Volver”.
5. Se muestra el menú principal.

Caso de Uso 2. Comenzar partida en modo arcade

Escenario principal

1. Se muestra el menú principal.
2. El jugador selecciona la opción “Jugar”.
3. Se muestra el menú de selección de modo de Juego.
4. El jugador selecciona la opción “Arcade”.
5. Se inicia el juego.

Escenario alternativo

1. Se muestra el menú principal.
2. El jugador selecciona la opción “Jugar”.
3. Se muestra el menú de selección de modo de Juego.
4. El jugador selecciona la opción “Volver”.
5. Se muestra el menú principal.

Caso de Uso 3. Pausar partida

Escenario principal

1. Se muestra el desarrollo normal de una partida.
2. El jugador presiona el botón de pausa.
3. Se muestra el menú de pausa.

Caso de Uso 4. Ver menú principal

Escenario principal

1. Se muestra el desarrollo normal de una partida.
2. El jugador presiona el botón de pausa.
3. Se muestra el menú de pausa.
4. El jugador selecciona la opción “Salir”.
5. Se muestra el menú principal.

Escenario alternativo

1. Se muestra el desarrollo normal de una partida.
2. El jugador presiona el botón de pausa.
3. Se muestra el menú de pausa.
4. El jugador selecciona la opción “Volver”.
5. Se muestra el desarrollo normal de una partida.

Caso de Uso 5. Ver opciones

Escenario principal

1. Se muestra el menú principal.
2. El jugador escoge la opción “Opciones”.
3. Se muestra el menú de opciones.

Caso de Uso 6. Ver créditos

Escenario principal

1. Se muestra el menú principal.
2. El jugador escoge la opción “Créditos”.
3. Se muestran los créditos.

Caso de Uso 7. Cambiar opción

Escenario principal

1. Se muestra el menú de opciones.
2. El jugador se desplaza por el menú entre las opciones disponibles hasta que llega a la que quiere cambiar.
3. El jugador cambia el valor de la opción.

Escenario alternativo

1. Se muestra el menú de opciones.
2. El jugador selecciona la opción “Volver”.
3. Se muestra el menú principal.

Caso de Uso 8. Disparar proyectil

Escenario principal

1. Se muestra el desarrollo normal de una partida.
2. El jugador presiona el botón de disparo.
3. Se dispara un proyectil.

Caso de Uso 9. Acelerar movimiento

Escenario principal

1. Se muestra el desarrollo normal de una partida.
2. El jugador presiona el botón de aceleración.

3. Se acelera la nave.

Caso de Uso 10. Decelerar movimiento

Escenario principal

1. Se muestra el desarrollo normal de una partida.
2. El jugador presiona el botón de freno.
3. Se decelera la nave.

Caso de Uso 11. Ver récords

Escenario principal

1. Se muestra el menú principal.
2. El jugador escoge la opción “Records”.
3. Se muestran los créditos.

Caso de Uso 12. Salir del juego

Escenario principal

1. Se muestra el menú principal.
2. El jugador selecciona la opción salir.
3. Se sale de la partida.

4.2 Diagrama de clases

A continuación se muestra un diagrama de clases, separado por módulos, debido a que el juego está separado en 3 grandes partes, y un pequeño diagrama de cómo los módulos se relacionan entre sí.

4.2.1 Aplicación

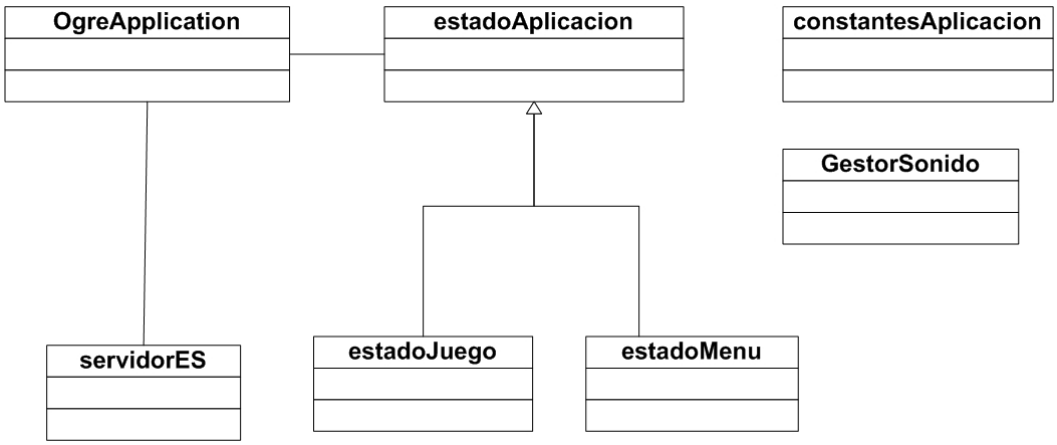

Figura 4.2. Diagrama de clases de la Aplicación

4.2.2 Gráficos

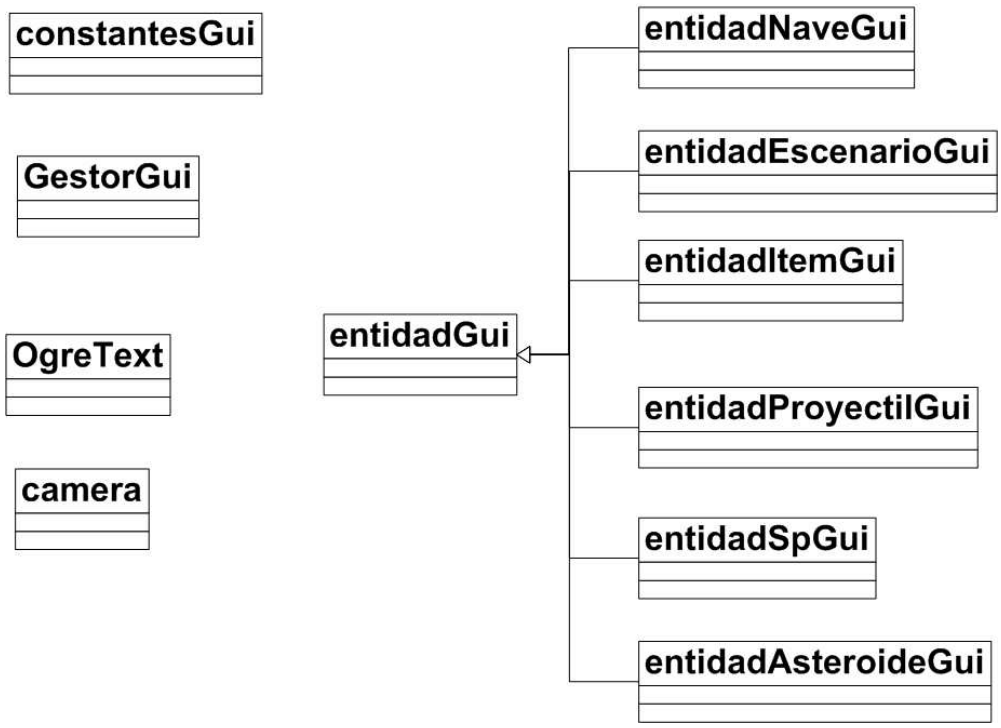

Figura 4.3. Diagrama de clases de la parte gráfica

4.2.3 Lógica

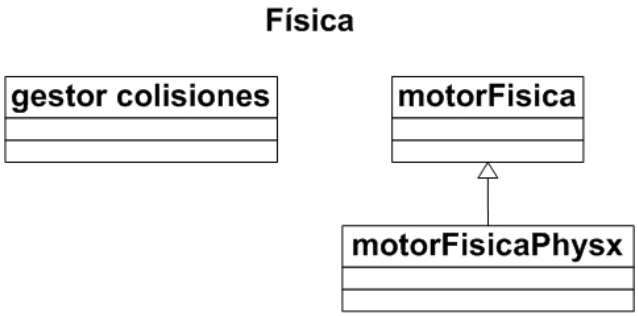

Figura 4.4. Diagrama de clases de la parte física de la lógica

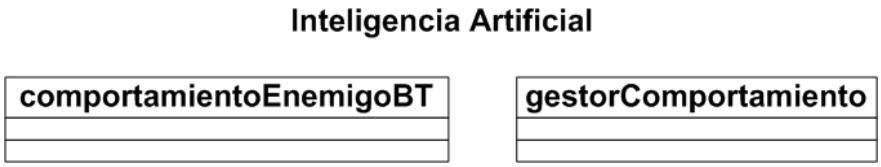

Figura 4.5. Diagrama de clases de la inteligencia artificial

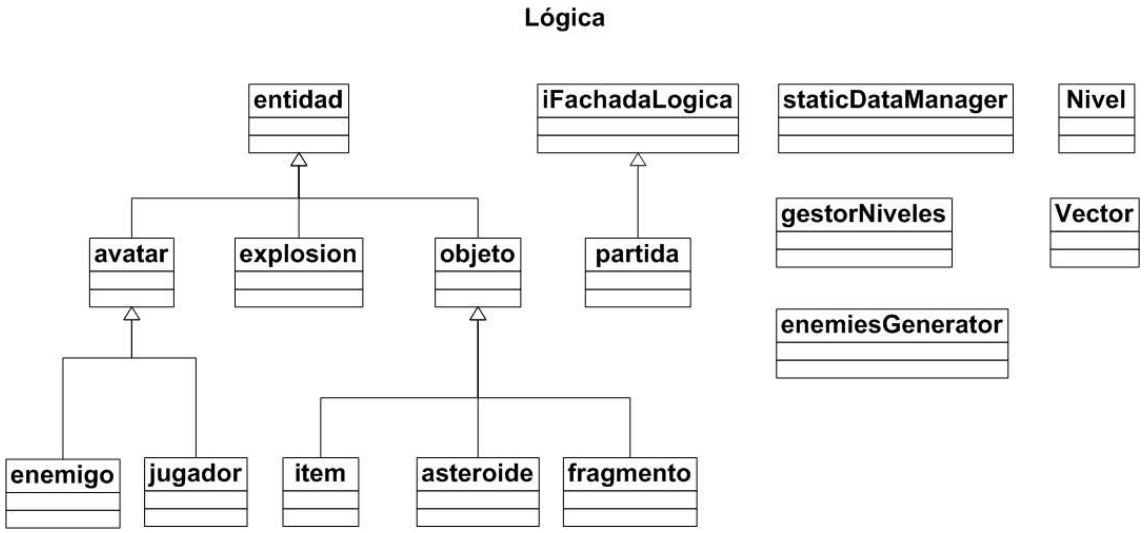

Figura 4.6. Diagrama de clases de la parte lógica

4.2.4 Relaciones entre módulos

A continuación se muestra un diagrama de clases donde se muestran las relaciones que existen entre las clases de distintos módulos.

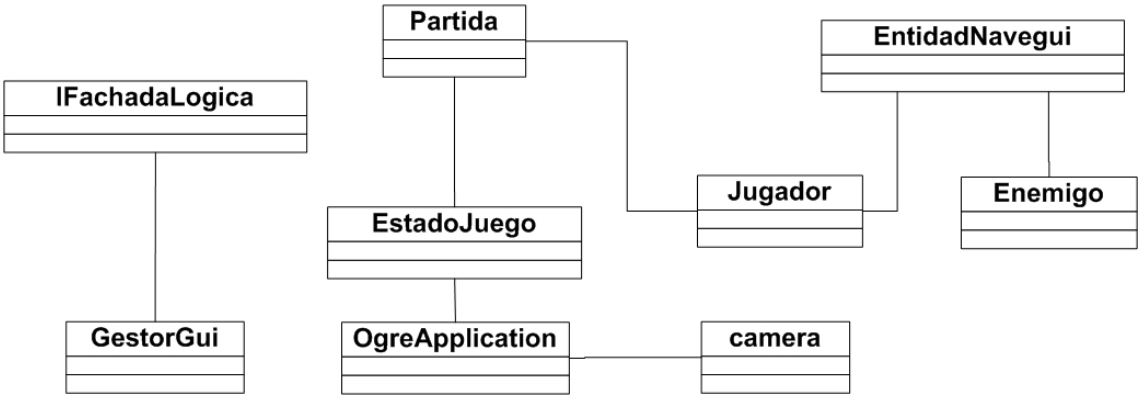

Figura 4.7. Diagrama de clases de relaciones entre módulos

Como se puede observar, existen dos partes, la de la izquierda entre la fachada lógica y el gestor de la parte gráfica, y la de la derecha entre la partida, el jugador y el enemigo, de la parte lógica, y clases de la parte de aplicación y gráficos.

Explicamos por orden en qué consisten estas relaciones:

- La fachada lógica implementa una clase de patrón observer de la partida, de la cual el gestor se hace observador. Esta clase implementa unas funciones de creación que serán llamadas cada vez que se cree una entidad lógica, para crear su entidad gráfica asociada.

- La partida tiene guardado el jugador, para avisarle de eventos importantes. A su vez, guarda una lista con todas las entidades de la partida, entre ellas los enemigos, para avisarles de que se tienen que actualizar. La partida tiene un método para actualizar las entidades

que es llamado por el estadoJuego. Este estado contiene la partida y le avisa de que tiene que actualizarse. La clase OgreApplication lleva las funciones de OGRE, que son las que avisan de cada tick de la aplicación. A su vez, guarda la cámara para tenerla accesible a toda la aplicación. La clase Avatar, de la que heredan el jugador y el enemigo, tiene una clase observer, de la cual se hace observadora la clase entidadNaveGui. Cada vez que cambia la energía o de visibilidad, la entidad gráfica actúa en consecuencia.

Capítulo 5

Diseño

En este capítulo vamos a describir los elementos del juego que han sido diseñados. Comenzaremos hablando del diseño de los personajes, luego explicaremos los ítems del juego, mostraremos la interfaz del juego, los menús, explicaremos las principales mecánicas del juego, contaremos el desarrollo de los modos de juego, que son historia y arcade y, por último, explicaremos la ambientación del juego.

5.1 Personajes

Existen varios tipos de personajes en el juego, de dos tipos principales: naves y torretas. Explicamos primero las naves, indicando sus valores de resistencia, velocidad y fuerza con rango de 1 a 5, donde 1 es el valor más bajo y 5 el más alto.

Para explicarlo se muestra una tabla, en la que se describen estos elementos, indicando el rango en el que se mueve cada parámetro.

Elemento	Descripción	Rango
Resistencia	Es el factor de daño que la nave puede aguantar. A mayor valor, menos daño harán los proyectiles	1(mínimo)-5(máximo)
Velocidad	Indica la rapidez con la que se mueve la nave. Cuanto mayor es, más rápido se mueve la nave	1(mínimo)-5(máximo)
Fuerza	Valor que marca el daño que hacen los proyectiles. A mayor valor, mayor daño.	1(mínimo)-5(máximo)
Cadencia	Marca la rapidez de disparo de las torretas. Cuanto mayor es, más rápido dispara.	1(mínimo)-3(máximo)

Tabla 5.1. Elementos de los personajes.

o Personaje principal

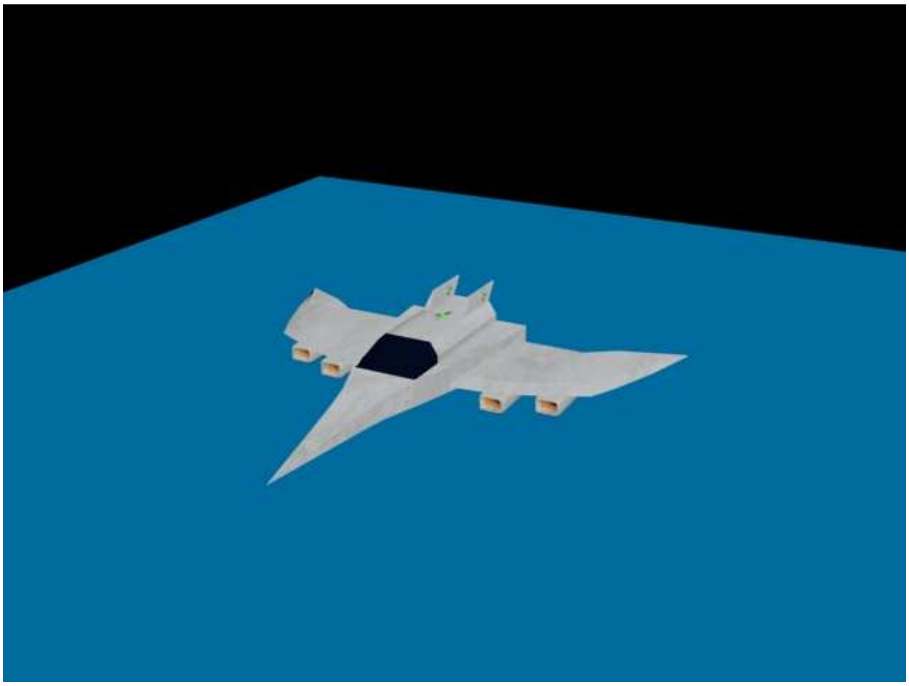

Figura 5.1. Nave principal.

El personaje principal, el que lleva el jugador, lleva una nave espacial estándar. Con la cámara seguiremos su recorrido, con vista en tercera persona, detrás de la nave. Los atributos de esta nave son variables, ya que el jugador puede recoger ítems que aumenten su potencia, además de poder aumentar su velocidad.

Resistencia: 2.

Velocidad: 2 - 3.

Fuerza: 2 - 4.

o Enemigo simple

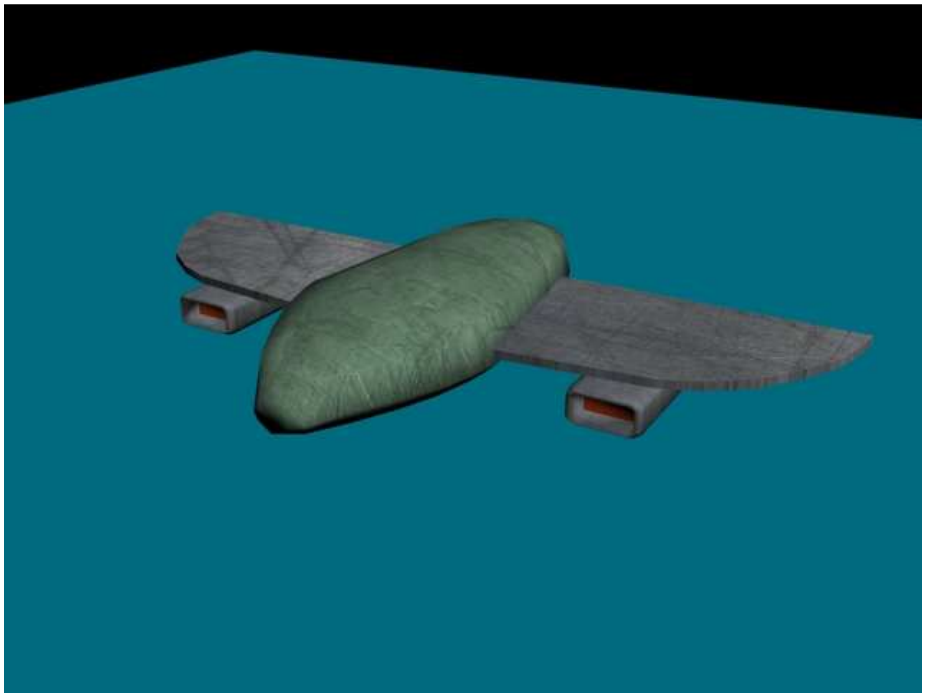

Figura 5.2. Enemigo simple

El enemigo simple es una nave espacial pequeña, muy rápida pero muy débil y hace muy poco daño. Su comportamiento es muy sencillo: cuando ve al jugador se dirige hacia él, apunta y dispara ráfagas de 3 disparos. Luego lo esquiva y desaparece.

Resistencia: 1.

Velocidad: 4.

Fuerza: 1.

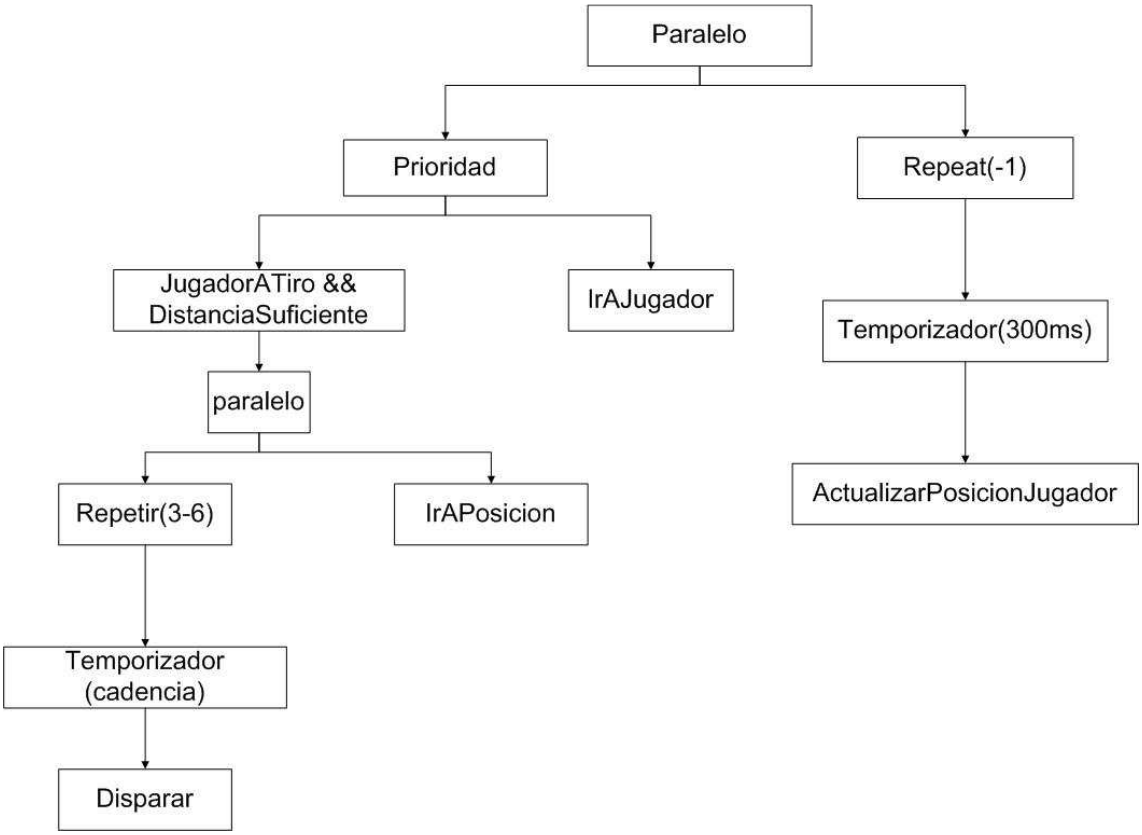

Figura 5.3. Árbol de comportamiento de las naves.

o Enemigo medio

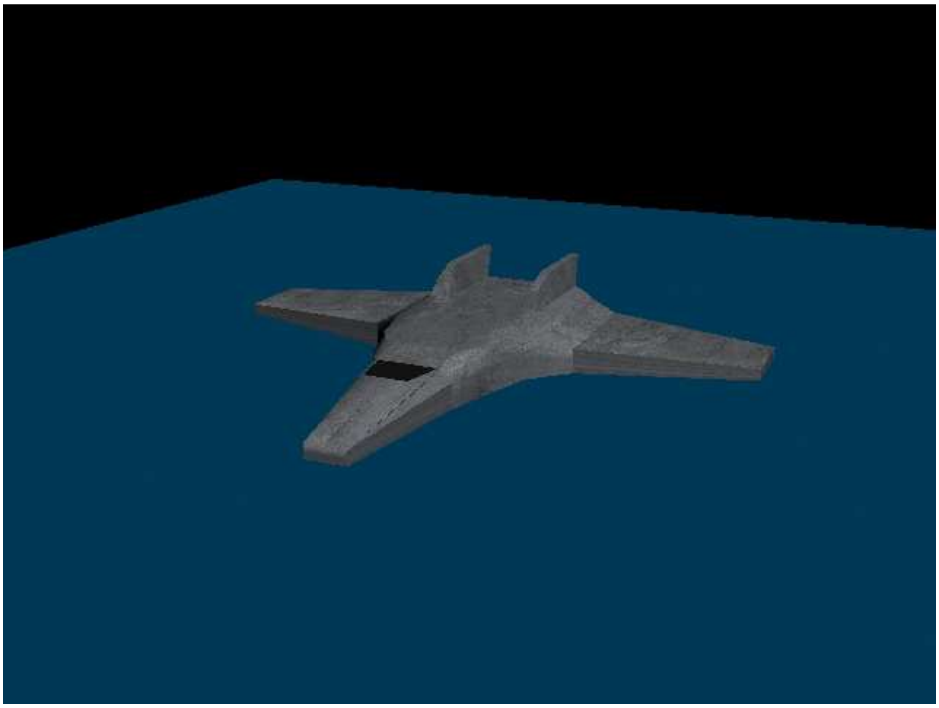

Figura 5.4. Enemigo medio

El enemigo medio es el enemigo estándar del juego, puesto que presenta una dificultad media. Es el más abundante del juego. Su comportamiento es el mismo que el enemigo simple.

Resistencia: 3.

Velocidad: 2.

Fuerza: 2.

o Enemigo fuerte

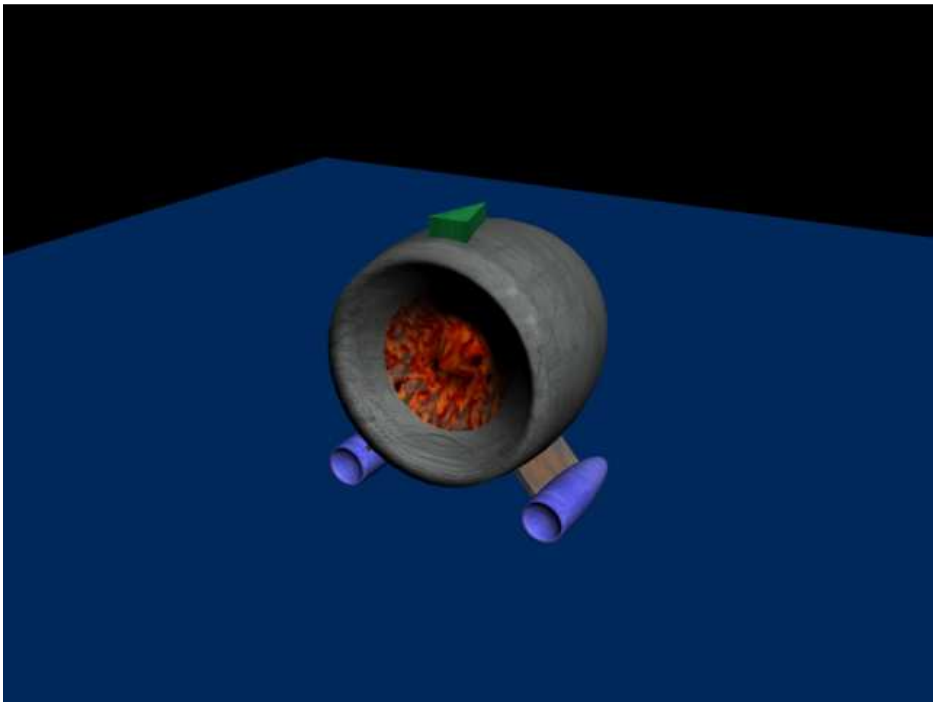

Figura 5.5. Enemigo fuerte

El enemigo fuerte es muy fuerte y resistente, pero muy lento. Además tarda bastante en disparar, pero hace mucho daño. Su comportamiento consiste en ir en línea recta y disparar al jugador cuando lo tenga a tiro.

Resistencia: 4.

Velocidad: 1.

Fuerza: 4.

0 Enemigo medusa

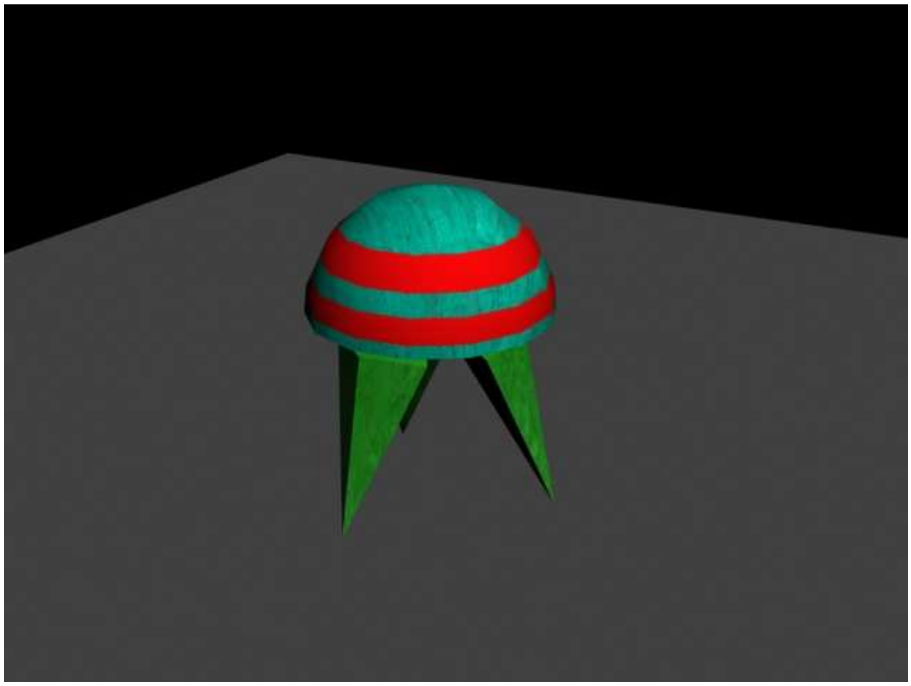

Figura 5.6. Enemigo medusa

El enemigo medusa es el más sencillo del juego, puesto que lo único que hace es intentar chocarse con el jugador. Dado su condición de trampas, su resistencia, velocidad y fuerza son bajas. Su comportamiento básicamente es ir de arriba hacia abajo, de ahí su nombre. No dispara proyectiles.

Resistencia: 1.

Velocidad: 2.

Fuerza: 0.

Pasamos ahora a describir las torretas. Las torretas son enemigos estáticos, que disparan sin parar al jugador en cuanto lo tienen a tiro. Todas las torretas tienen el mismo comportamiento, y la única diferencia entre ellas es el daño que hacen y la cadencia de tiro que tienen.

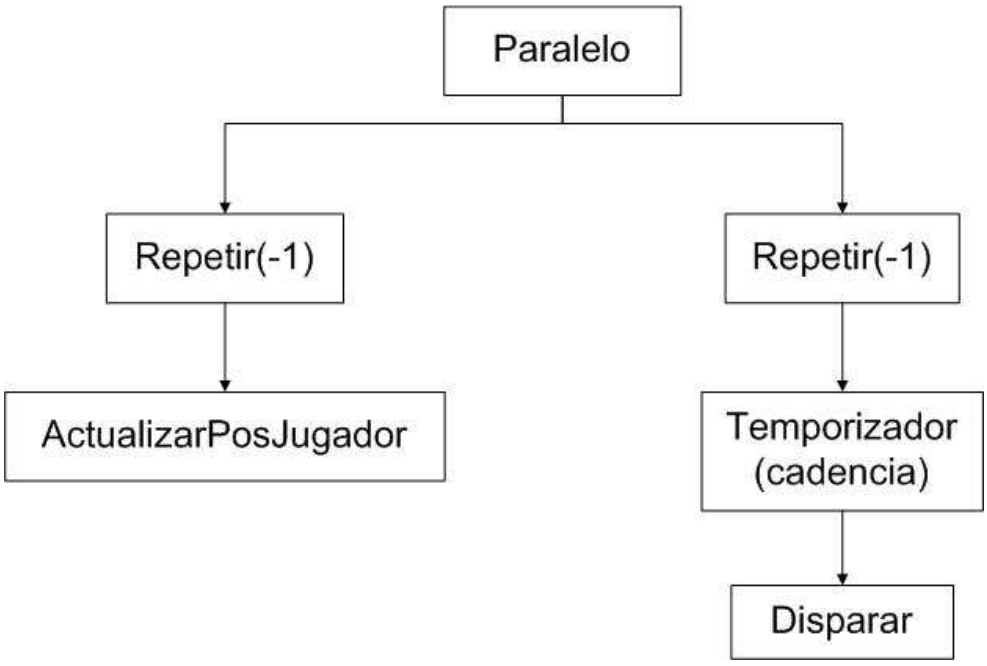

Figura 5.7. Árbol de comportamiento de las torretas

- Torreta de tierra 1

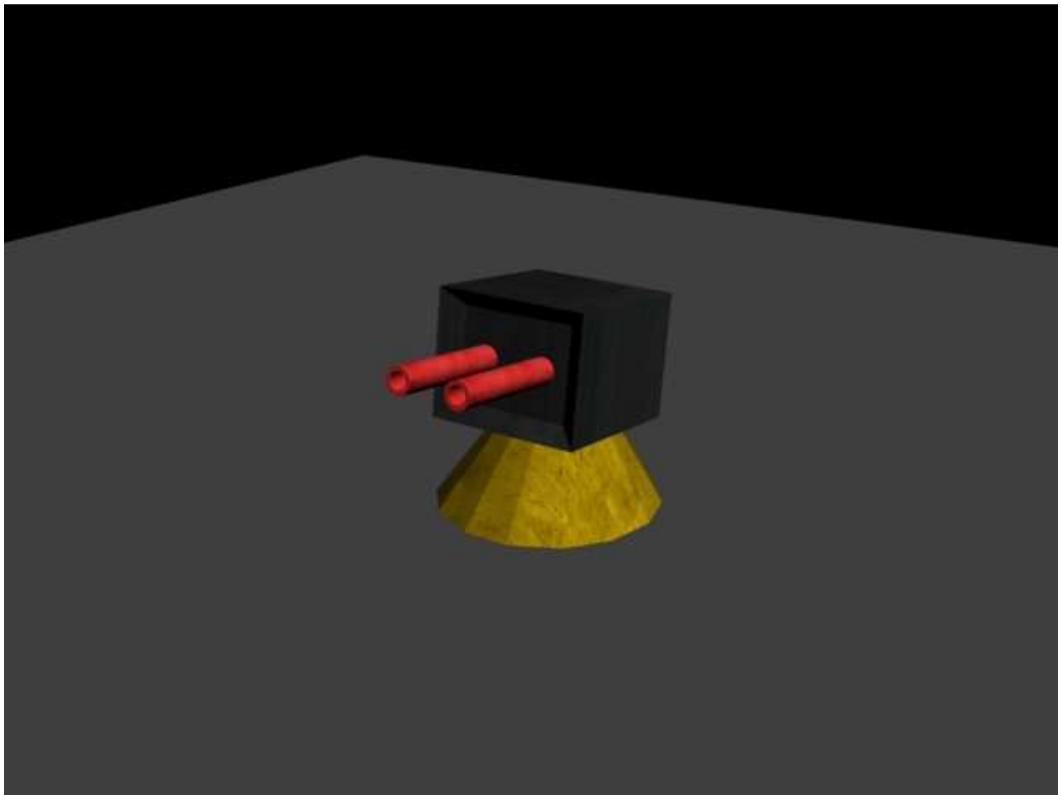

Figura 5.8. Torreta de tierra 1.

Potencia: 1

Cadencia: 1

0 Torreta de tierra 2

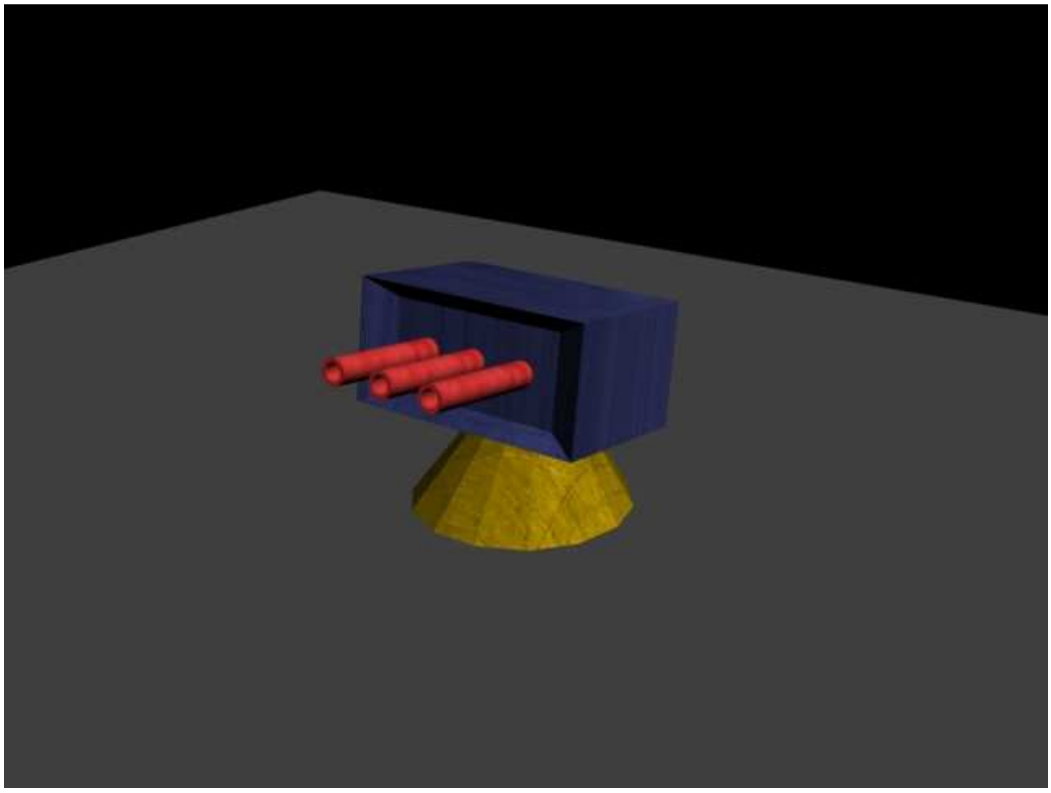

Figura 5.9. Torreta de tierra 2

Potencia: 2

Cadencia: 1

- Torreta de tierra 3

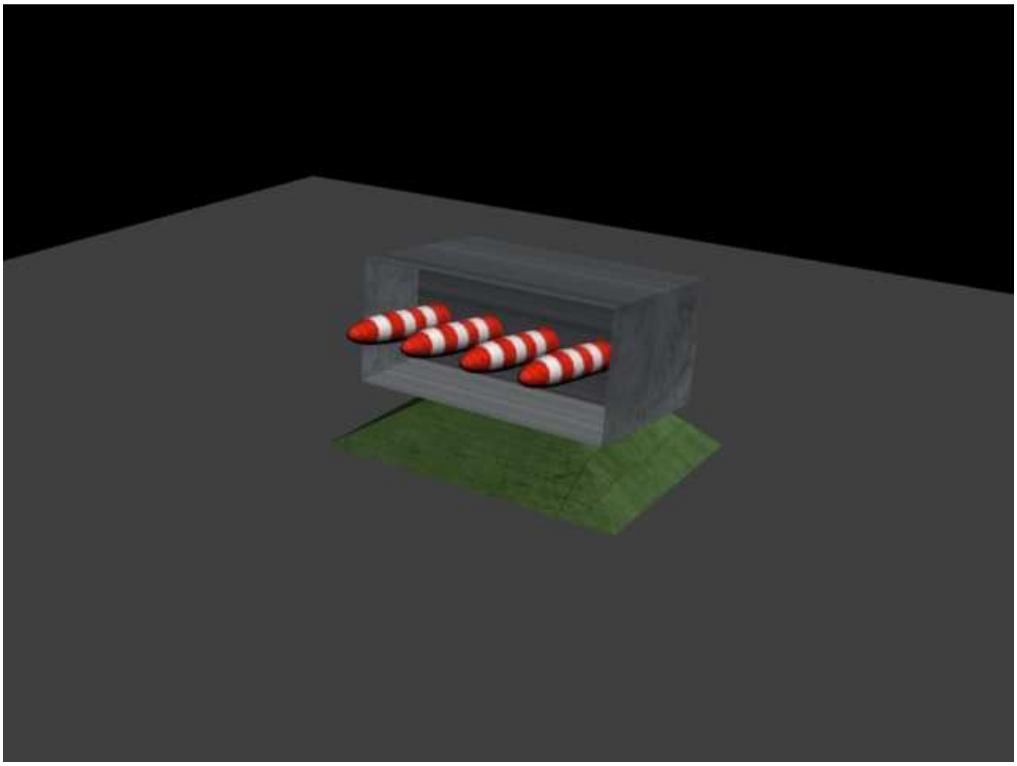

Figura 5.10. Torreta de tierra 3

Potencia: 3

Cadencia: 3

o Torreta de aire

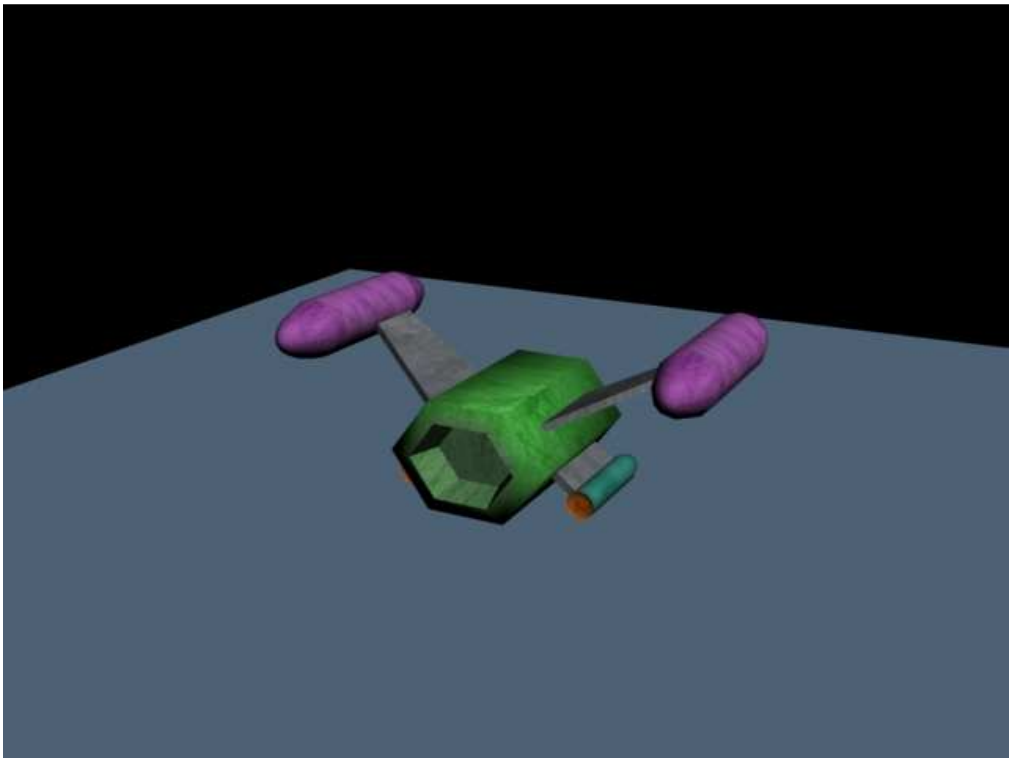

Figura 5.11. Torreta de aire

Potencia: 1 - 2

Cadencia: 2 – 3

La variación en las torretas de aire se debe a que en la primera fase son más débiles y en la segunda crece la dificultad.

5.2 Objetos

5.2.1 Items

- Puntos

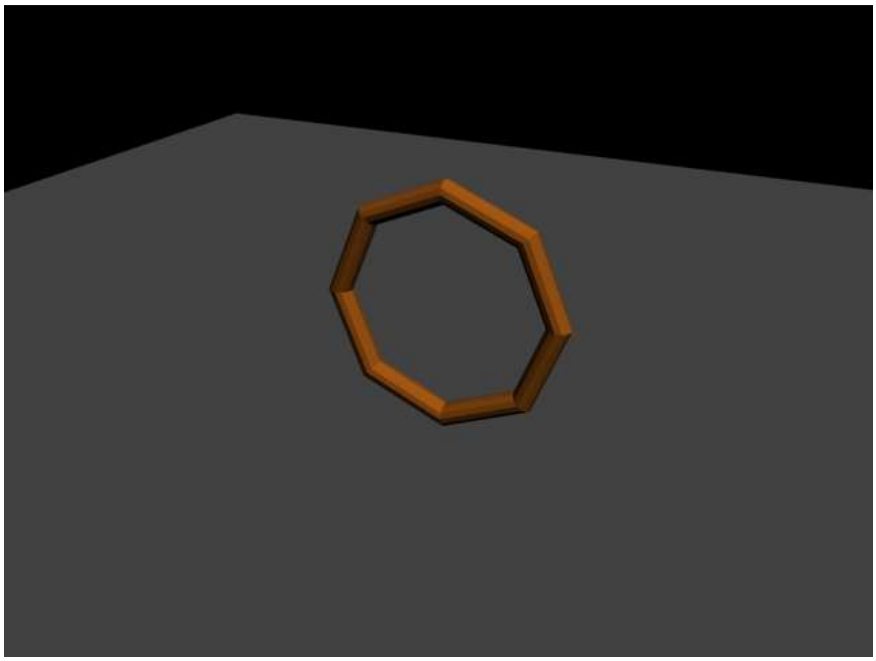

Figura 5.12. Puntos

Este ítem incrementa el número de puntos de la partida del jugador. Cuantos más consiga, más alto estará en la tabla de puntuaciones. Existen tres tipos: oro (500 puntos), plata (200 puntos) y bronce (100 puntos).

- Salud

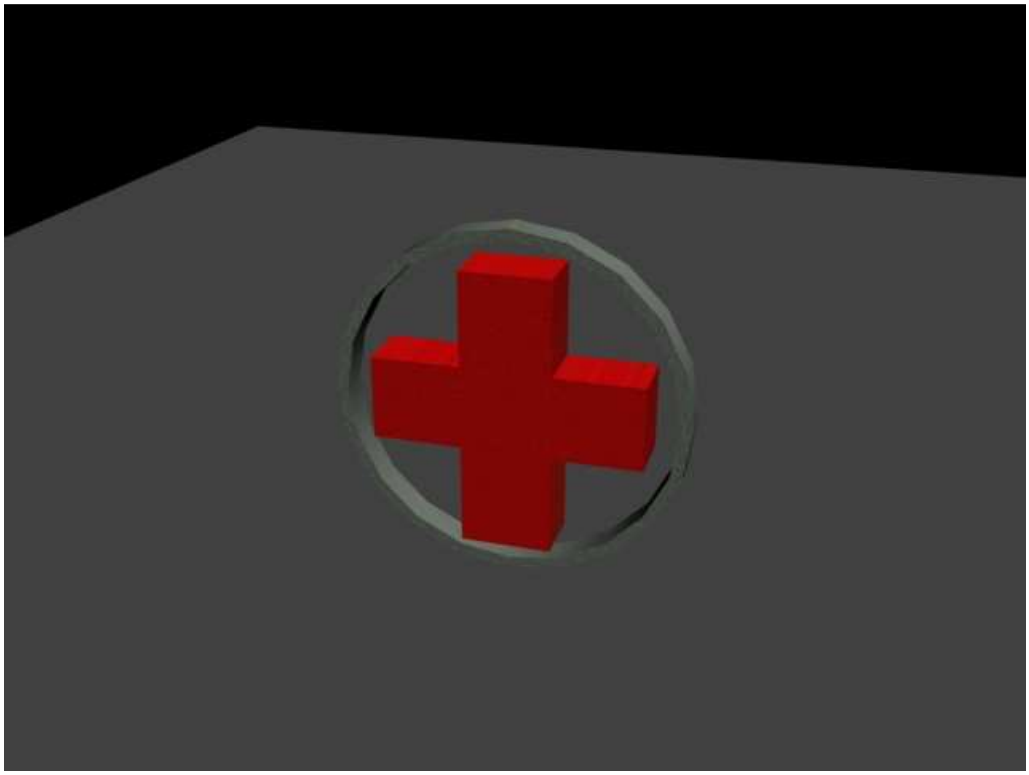

Figura 5.13. Salud

Este ítem restaura parcialmente la salud del jugador.

- Potencia

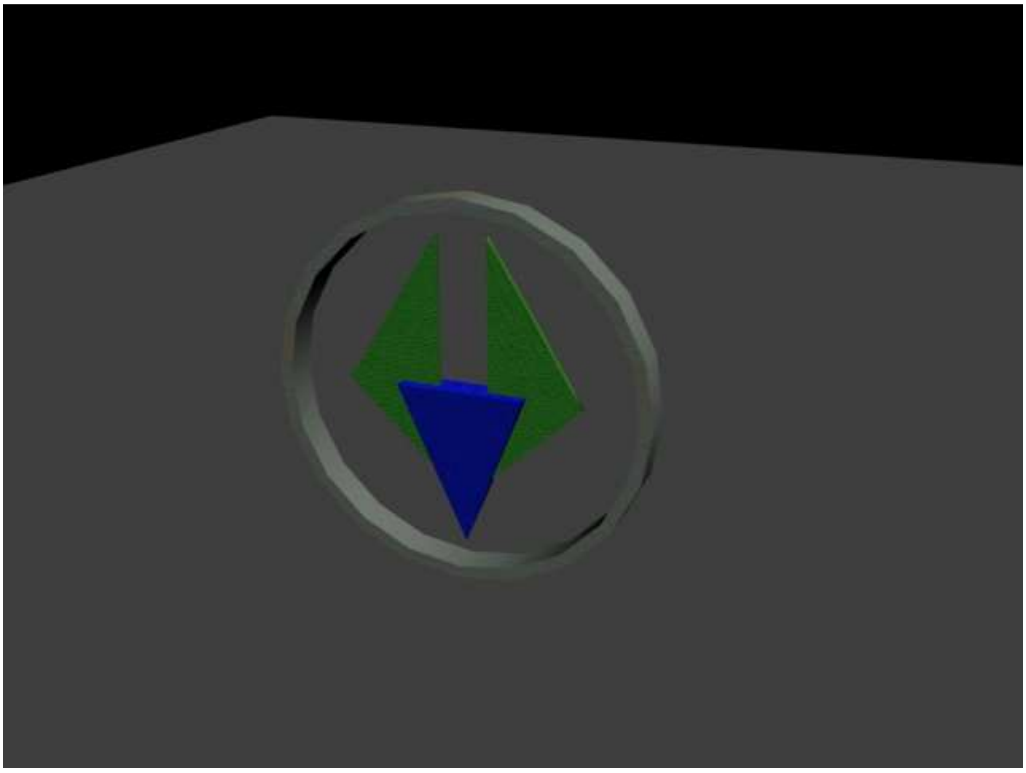

Figura 5.14. Potencia

Este ítem incrementa la potencia del disparo del jugador.

- Vida

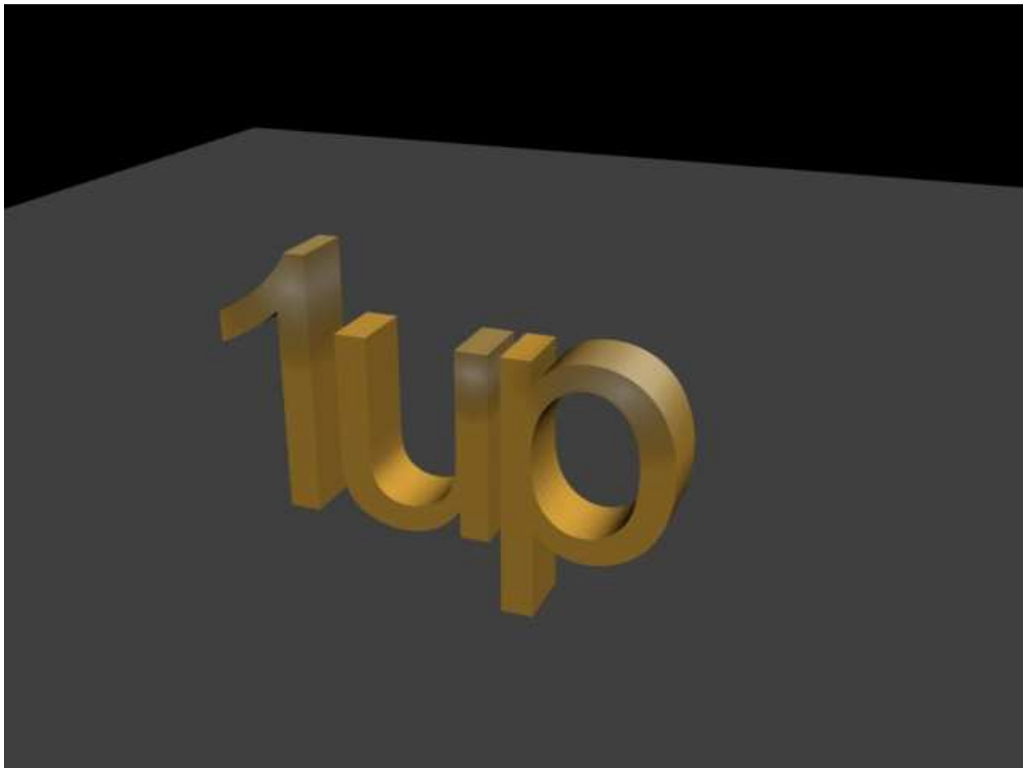

Figura 5.15. Vida

Incrementa en uno el número de vidas del jugador.

5.2.2 Proyectiles

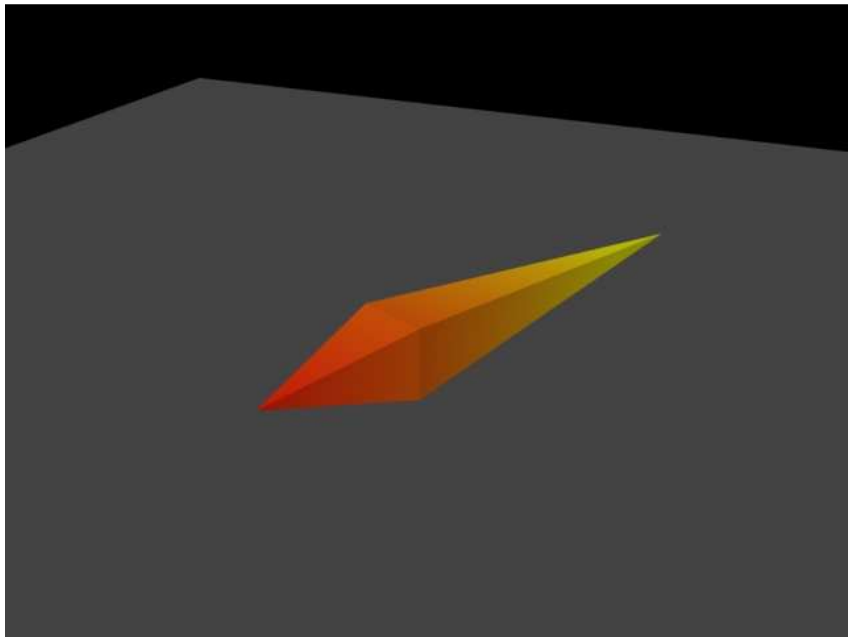

Figura 5.16. Proyectil

Los proyectiles de la partida. Son iguales para todas las naves, a excepción de la del jugador, que varía según la potencia, pasando por dos estados: medio y fuerte. El primero de color amarillo y rojo, y el segundo de color verde y azul intenso.

5.2.3 Asteroides

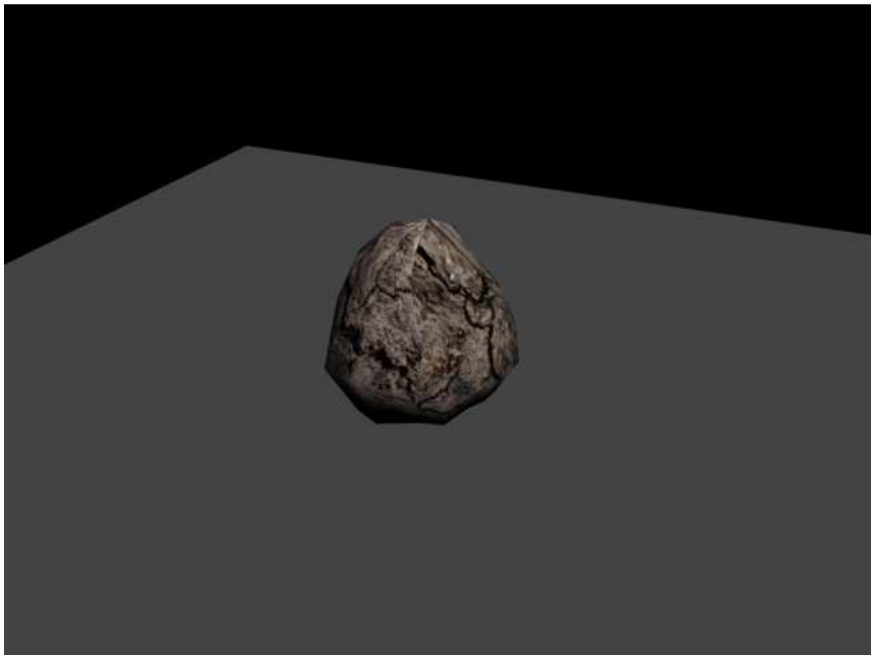

Figura 5.17. Asteroide

Los asteroides son rocas espaciales, restos de planetas explosionados, que flotan en el espacio con una trayectoria lineal y constante. El jugador puede explotarlos o esquivarlos.

5.3 Interfaz

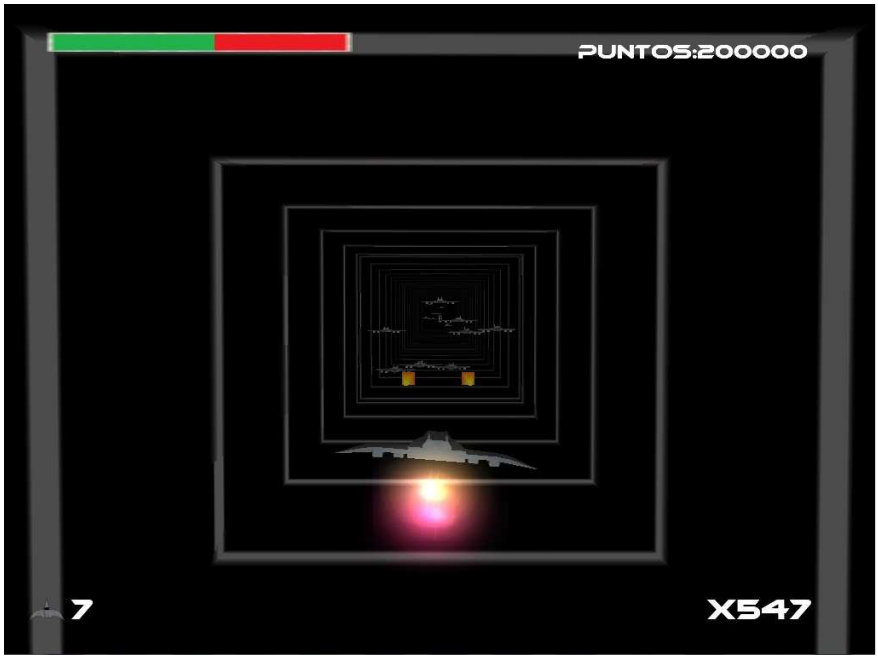

Figura 5.18. Interfaz del juego o HUD

Tenemos cuatro grupos importantes en la interfaz. Por una parte, la barra de energía del jugador, en la esquina superior izquierda, indica cuanta energía le queda para perder una vida. De color verde es la energía restante mientras que, según se va perdiendo, se va poniendo de color rojo.

En la esquina superior derecha, se muestra la puntuación que lleva el jugador en la partida. Este valor se va calculando según el número de naves que mate, el número de ítems de puntos que coja y los asteroides que destruya, que darán una pequeña cantidad de puntos.

En la esquina inferior derecha tenemos el número de naves destruidas de la partida. Este valor es simplemente indicativo, y se muestra en la pantalla de récords.

En la esquina inferior izquierda está el número de vidas pendiente que queda.

5.4 Menús

- Menú principal

El menú principal es el que sale nada más iniciar el juego. En él se nos presentan una serie de opciones que se indican a continuación:

Figura 5.19. Menú principal

- Jugar: Nos lleva al modo de selección de juego, que tiene tres opciones: Historia, Arcade y Volver. La primera nos inicia el modo Historia, la segunda inicia el modo Arcade y la tercera vuelve al menú principal.
- Opciones: Lleva al menú de opciones.
- Récords: Nos muestra los récords del modo historia y el modo arcade.
- Créditos: Nos muestra los créditos del juego.
- Salir: Sale del juego.

- Menú opciones

El menú de opciones sirve para cambiar el sonido y la resolución del juego. Las opciones que presenta son:

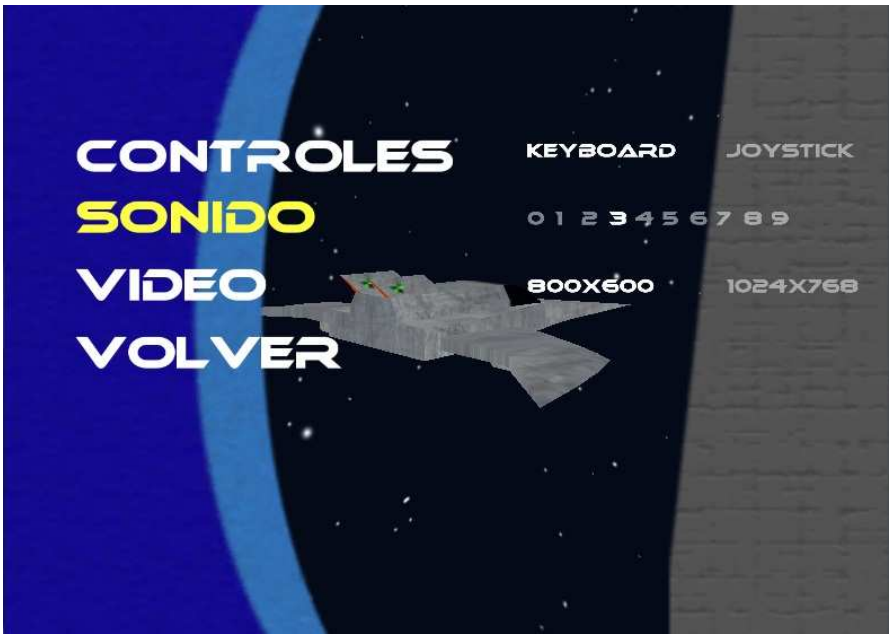

Figura 5.20. Menú de opciones

- Sonido: Cambia el volumen del juego.
- Resolución: Cambia la resolución del juego entre 800x600 y 1024x768.
- Volver: Vuelve al menú principal.

- Menú de pausa

El menú de pausa sirve para parar la partida y salir del juego. Las opciones disponibles son:

Figura 5.21. Menú de pausa.

- Volver al juego: Cierra el menú de pausa y vuelve a la partida.
- Salir del juego: Termina la partida actual y vuelve al menú principal.

- <u>Menú de Créditos</u>

Este menú muestra los créditos del juego, es decir, las personas que han trabajado en este proyecto.

Figura 5.22. Menú de créditos.

- Menú de Récords

Este menú muestra las puntuaciones más altas del juego del modo Historia y del modo Arcade.

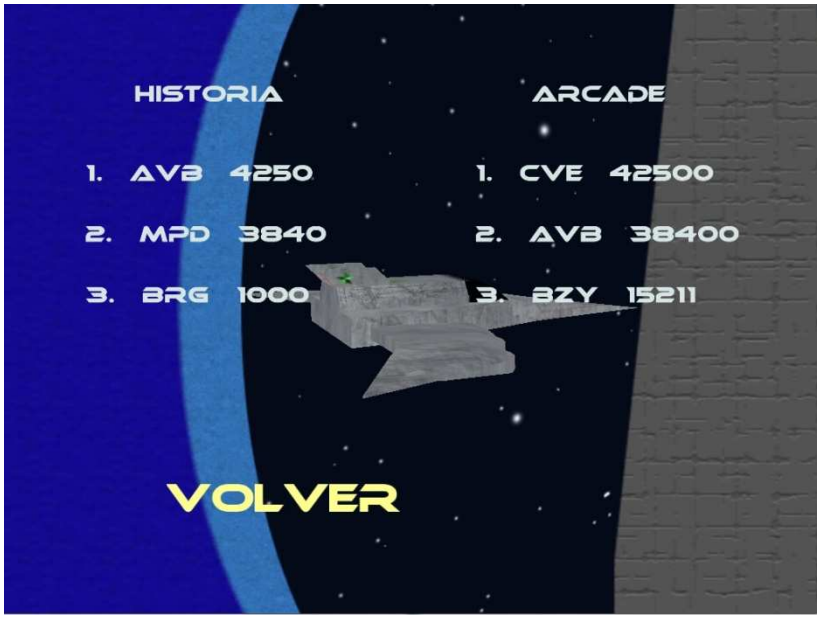

Figura 5.23. Menú de récords.

5.5 Mecánicas centrales

- Movimiento

La nave del jugador se mueve en línea recta por el eje de coordenadas Z a una velocidad constante. El jugador puede moverse con una limitación en altura y anchura libremente.

- Disparo

El jugador tiene una única arma que es el disparo de proyectiles. Se lanza de dos en dos y tiene diferente potencia, según los ítems que haya ido recogiendo.

- Aceleración y freno

El jugador puede acelerar la nave y frenarla a su antojo apretando el botón correspondiente. La aceleración y freno es sutil con respecto a su velocidad constante.

- Recoger ítem

El jugador puede recoger una serie de ítems a lo largo del escenario. Cada ítem le otorga un beneficio distinto, cuyos efectos son: recuperar energía, aumentar la potencia, aumentar los puntos de la partida o incrementar el número de vidas.

5.6 Modos de Juego

- Modo historia

El modo historia consta de dos fases desarrolladas en dos escenarios distintos. El objetivo del jugador es conseguir llegar al final de los dos escenarios, quedándole al menos la vida cero, con el mayor número de puntos posible. Si consigue el número suficiente de puntos entrara en el ranking de las mejores puntuaciones. La primera fase se desarrolla en el

espacio y abundan los asteroides, las naves simples y medias y las torretas de aire. El segundo nivel se desarrolla en un planeta pequeño y aquí abundan las naves medias, fuertes y las torretas de tierra y aire.

- Modo arcade

El modo arcade consiste en un escenario infinito, durante el cual se le van a aparecer al jugador las naves enemigas aleatoriamente en función de la vida que tengamos, el número de puntos conseguidos y la distancia recorrida, de manera que cada jugador consiga una partida acorde a su nivel de destreza con el juego. Cada cinco tandas de enemigos aparecerá un ítem aleatorio que también dependerá de la vida, la potencia y los puntos que lleve.

5.7 Ambientación

Estamos en el año 2457 d.C. Las tropas intergalácticas estelares dominan el comercio de toda la galaxia. El ejército Xel' thalá de la galaxia UB543b ha conseguido llegar a esta galaxia en pos de conseguir el objetivo de dominar el universo. Con lo que no contaban es que su mayor guerrero estaba fuera en misión de reconocimiento. Cuando vuelve de su misión encuentra que las tropas enemigas están arrasando su planeta y no lo puede permitir…

Capítulo 6

Implementación

A la hora de implementar el juego intervienen las siguientes partes: código, arte 2D, arte 3D y sonido. El código es el programa que se encarga de manejar el juego, moviendo las entidades 3D por el escenario, gestionar la entrada del usuario y actuar en consecuencia. Para implementar el código se ha utilizado el sistema Microsoft Visual Studio que tiene una integración con las librerías de OGRE muy optimizada, además del proceso de vinculación con DirectX.

Entendemos por arte 2D todo aquello que es una imagen plana en el juego, esto es, las texturas del menú y las texturas de las entidades 3D. Estas se pueden realizar con cualquier programa de creación de imágenes, como pueden ser OpenOffice Draw, Gimp o Photoshop. Concretamente para este proyecto se han utilizado Gimp y Photoshop.

Los modelos 3D son las entidades del juego que representan un volumen en el espacio, pero sin gráficos, sólo un montón de vértices unidos por aristas conectadas. Un grupo de aristas conectadas forma un plano que, para simplificar cálculos de la GPU, se hace triángulos. Luego al conjunto de planos se les aplica un mapeado de coordenadas de vértices, comúnmente llamado plano de UV's, el cual es un plano 2D en el que se definen una posición para cada vértice, que se traduce en una coordenada de textura para, por último, trasladarlo a la malla y pintar sobre ella la textura. Los modelos 3D se pueden realizar con muchos

programas, por ejemplo, Blender, Maya o 3ds Max. En este proyecto se ha utilizado 3ds Max por motivos que se verán más adelante.

Para gestionar el sonido se ha utilizado un motor comercial que permite una licencia gratuita si es para uso no comercial. El motivo principal de escoger este motor y no otro, como SDL Mixer, es que ya he utilizado el motor en otro proyecto, obteniendo muy buen resultado.

A continuación, se presenta un diagrama de flujo de datos, en el cual, se muestra el flujo del bucle principal del juego, que se llama en cada frame de la aplicación. OGRE avisa en cada frame de que va a pintar, la aplicación recoge ese evento y llama al estado actual que se actualice. En el estado del juego, llama al servidor de Entrada/ Salida para recoger los eventos del jugador, después llama al gestor de los gráficos para que las entidades gráficas se pinten y, por último, llama a actualizar la partida. La partida procesa los comandos que tenga almacenados, manda actuar en consecuencia a jugador y enemigos, el motor de física simula lo que ocurre, manda actualizar a todas las entidades lógicas de la partida, las cuales, insertan eventos (muerte, cambio de segmento, etcétera…), actualiza la cámara y procesa los eventos pendientes.

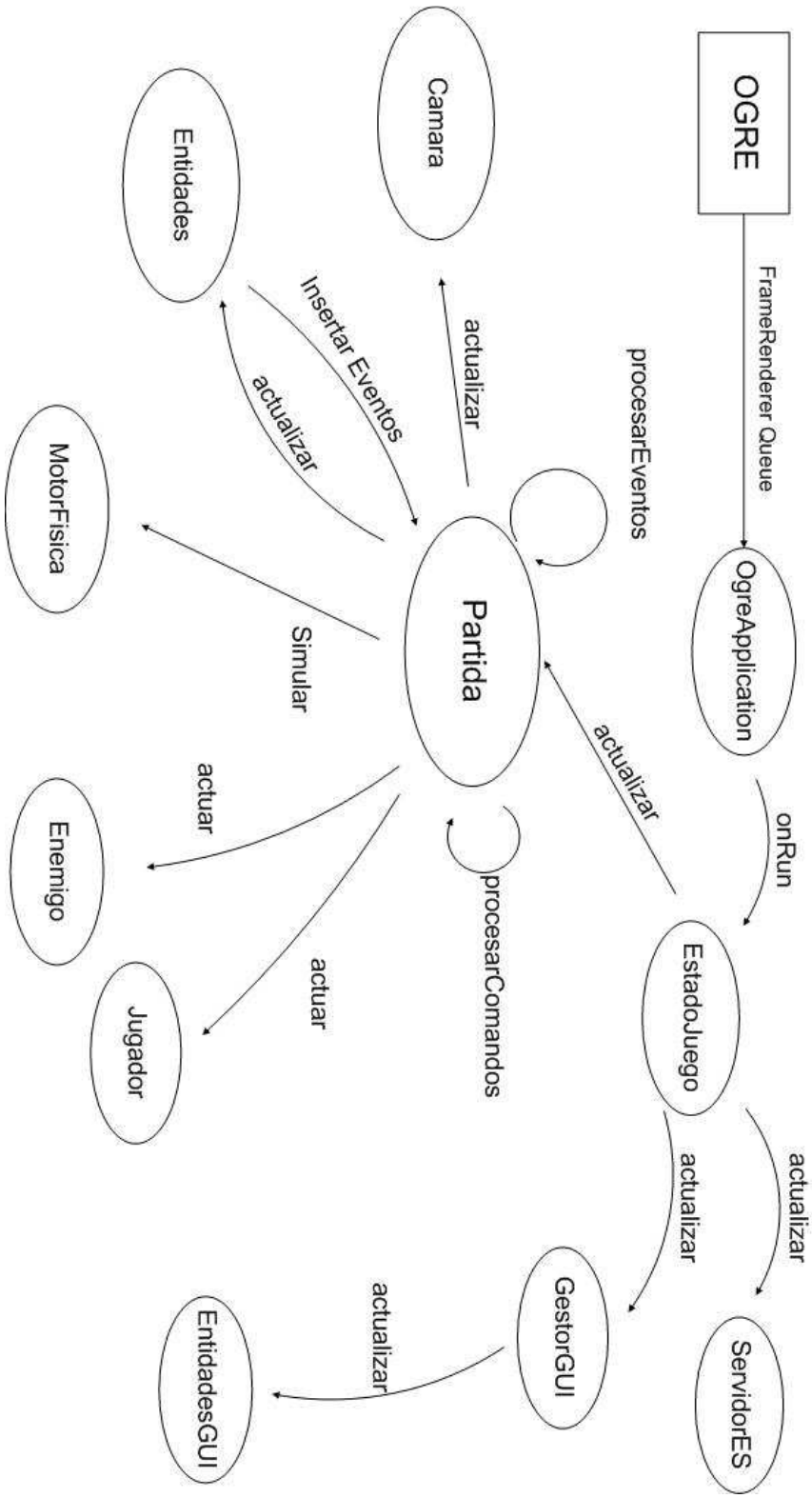

Figura 6.1. Diagrama de Flujo de Datos del bucle principal.

6.1 Código

La programación podemos dividirla en varios módulos claramente diferenciados:

6.1.1 Aplicación

Para manejar la aplicación se ha optado por implementar una máquina de estados, donde cada una implementa el método onRun() que se llamara en cada tick entre pintado de frames.

En este módulo se incluyen todas aquellas clases que manejan el desarrollo de la aplicación. Las clases más relevantes son:

- *OgreApplication*: Es la clase encargada de inicializar y gestionar los recursos de Ogre, además de gestionar la máquina de estados de la aplicación.
- *estadoAplicacion*: Es la clase padre de la máquina de estados, cada estado debe implementar los métodos que se definen.
- *estadoJuego*: El estado principal del juego, tiene dos modos, Arcade e Historia, según el modo que el usuario elija en el menú principal.
- *estadoMenu*: Este estado implementa los menús iniciales de la aplicación. Existen 3 menús principales, el inicial, el de opciones y el de juego.
- *servidorES*: Esta clase es una capa de abstracción del método de entrada del usuario, en función de si prefiere joystick o teclado.
- *gestorSonido:* Se encarga de actuar de interfaz entre el motor de sonido y la aplicación.

6.1.2 Gráficos

Para los gráficos se ha optado por crear un gestor que implementa un patrón observer de la clase partida de la lógica.

- *camera*: Esta clase implementa una capa por encima del manejador estándar de cámaras de OGRE, para añadir funcionalidades y

abstracción al manejador. Ha implementado un método update, en el cual, el estado de la aplicación que use la cámara, puede hacer lo que quiera con ella, como por ejemplo el estado menú que gira en círculos alrededor de la nave protagonista.

- *entidadgui*: Clase que define los métodos que deberán implementar las entidades gráficas que sean manejadas por el gestor.
- *GestorGui*: La clase principal de éste módulo. Se encarga de crear cada entidad gráfica del juego, así como de gestionar los recursos y liberarlos cuando dejan de ser útiles.
- *entidadnavegui*: Entidad gráfica que define la visualización de las naves.
- *entidaditemgui*: Entidad gráfica que define la visualización de los ítems.
- *entidadspgui*: Entidad gráfica que define la visualización de los sistemas de partículas.
- *entidadproyectilgui*: Entidad gráfica que define la visualización de los proyectiles.
- *entidadescenariogui*: Entidad gráfica que define la visualización de los escenarios.
- *OgreText*: Clase que gestiona la impresión de texto en pantalla. Implementa un patrón singleton para poder ser accesible desde todo el código, ya que es necesario en muchos puntos del código imprimir texto, como por ejemplo en la clase partida para mostrar las vidas o los puntos, o en el estado menú para imprimir las opciones.

6.1.3 Lógica

La lógica del juego es la parte más importante, ya que se encarga de gestionar todo el juego. Desde mover los personajes, calcular las colisiones o manejar los datos del juego. Podemos dividirlo en 3 grandes módulos: física, inteligencia artificial y lógica, propiamente dicha.

- **Física**

La física se implementa mediante una librería externa llamada PhysX. PhysX proporciona una gran abstracción de los cálculos físicos ya que es un motor muy optimizado que se encuentra en numerosos videojuegos comerciales, como Batman Arkham Asylum o Metro 2033. Para poder usarlo se definen las siguientes clases:

- *GestorColisiones*: Para que PhysX te informe de las colisiones, puedes implementar un gestor, que sobrescribe ciertos métodos, de manera que cuando se produce un choque entre dos entidades te avisa para que puedas hacer lo que quieras. Según que entidades hayan colisionado, tomaremos unas decisiones u otras.
- *MotorFisica*: Clase que implementa un singleton para abstraernos de la implementación del motor de física que utilicemos. Se hizo de esta manera puesto que si en un futuro se quisiera cambiar de motor de física a otro libre, o implementar el nuestro propio, no habría que cambiar el resto del código.
- *MotorFisicaPhysx*: Implementa las funciones que se definen en el Motor de Física, mediante las llamadas a las funciones de PhysX.

PhysX cuenta con una herramienta muy útil en el desarrollo de aplicaciones que utilicen este motor. Proporciona un debugger que muestra en una ventana todas las entidades físicas que se encuentran en la escena. Cuando lo ejecutamos sale esta ventana:

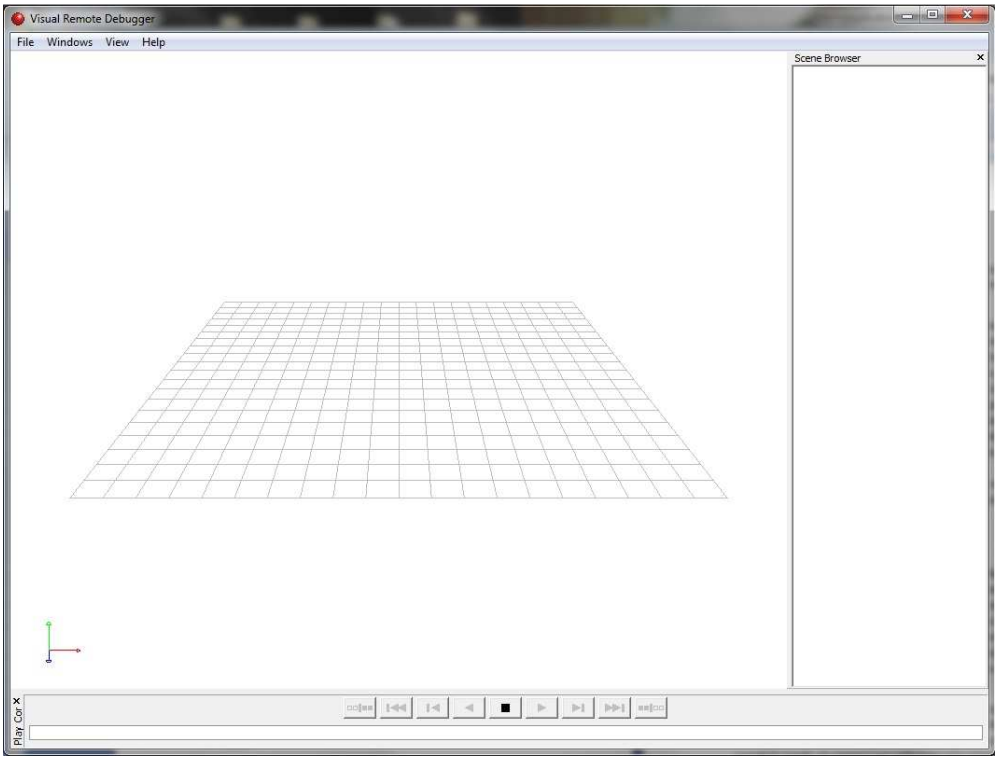

Figura 6.2. Ventana de PhysX Debugger Vacía.

Que básicamente no muestra nada porque no tenemos nada en la escena. Cuando iniciamos el programa, tenemos que crear una conexión remota a través de un puerto para conectarnos con el debugger. Y PhysX enviará los comandos y los mostrará en la ventana, quedando algo como esto:

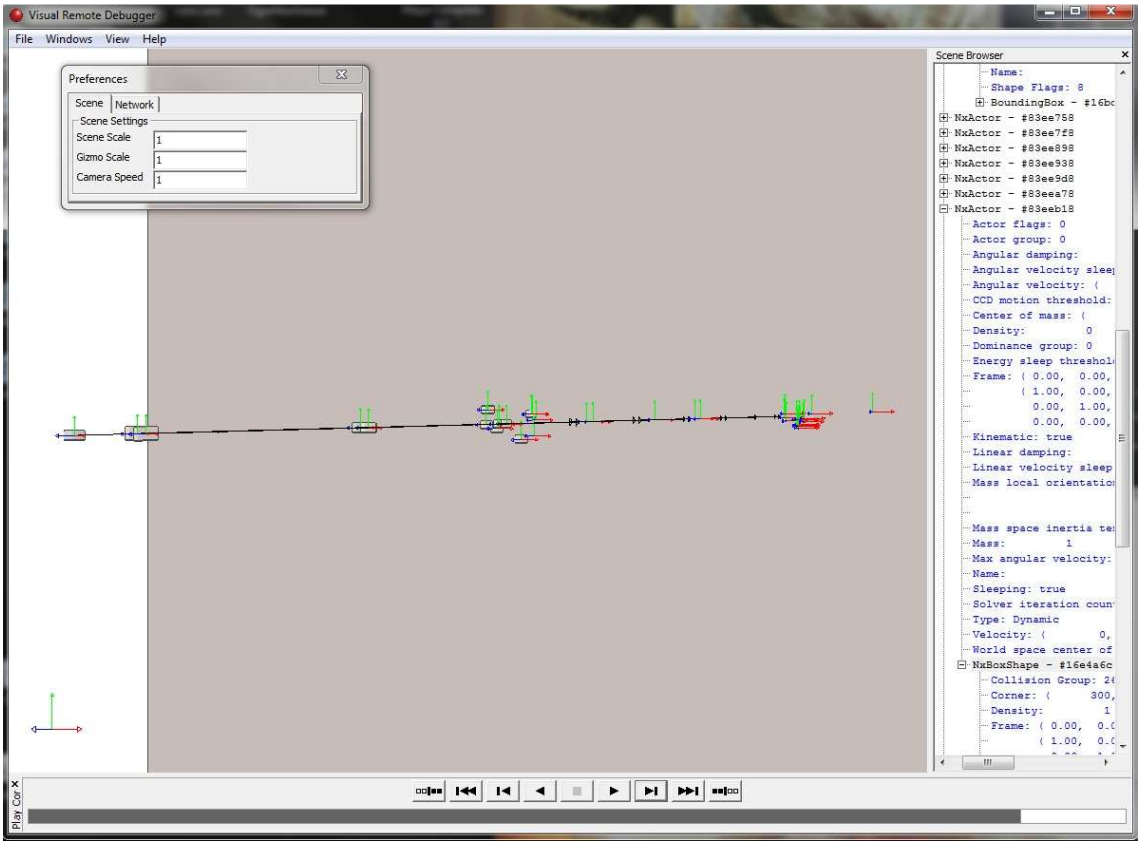

Figura 6.3. PhysX debugger en acción

En la parte izquierda podemos ver la escena, representada por vectores, cajas y esferas que poco tienen que ver con los gráficos del juego, pero que es la representación interna del motor físico. A la derecha, en letras de color azul y negro, se muestra la información de los valores de las entidades, tales como fuerza, orientación, masa, velocidad, etcétera... Y abajo vemos dos pequeñas barras, la de más abajo es la barra de tiempo y la que está encima es la barra de control del tiempo. Haciendo click en la barra de control podemos pasar hacia adelante, hacia atrás, pausar, ir al principio, ir al final o ir hacia adelante y hacia atrás frame a frame. Haciendo click sobre la barra de tiempo podemos ir a cualquier instante concreto de la ejecución.

- **Inteligencia Artificial**

La inteligencia Artificial se ha implementado mediante Árboles de comportamiento. [MdvUcm] Un Árbol de comportamiento es una estructura de datos con forma de árbol, donde cada uno de los nodos define un patrón de comportamiento. El código de los árboles de comportamiento se ha sacado de un proyecto anterior, en el que se definieron varios comportamientos que no sirven para éste al tener mecánicas distintas. Pero la estructura de los Árboles se ha podido reutilizar, cambiando los nodos.

Las clases principales son:

- *Módulo base*: En este módulo tenemos las clases base que nos van a ayudar a definir los comportamientos. Tenemos la clase BehaviorTreeBase, que define las clases básicas como son los nodos base y las condiciones de comprobación, y las clases de los Nodos básicos, que son:
  - Nodo paralelo: Hace que se ejecuten sus hijos en paralelo.
  - Nodo secuencial: Hace que se ejecuten sus hijos en el orden en que se han hecho hijos.
  - Nodo repetir: hace que se repita su hijo n veces.
  - Nodo probabilidad: hace que se ejecuten sus hijos con una probabilidad arbitraria.
  - Nodo prioridad: interrumpe la ejecución, para que su hijo se ejecute, cuando se cumple su condición.

- *Módulo de Nodos*:
  - NodeActualizarPosJugador: Actualiza la posición conocida del jugador en el enemigo.
  - NodeBooleanConditionWithChildren: Se inicializa con una condición de partida, por ejemplo, el jugador está a pocos metros, y se ejecutan sus hijos si se cumple la condición.

    - NodeDispararAJugador: Dispara a la última posición conocida del jugador, con una cierta puntería aleatoria.
    - NodeIrAPunto: Dirige al enemigo a un punto establecido.
    - NodeIrAJugador: Hereda de IrAPunto para establecer como punto al que dirigirse, la última posición conocida del jugador.
    - NodePriorityInterrupt: Evalúa sus hijos en el orden insertado, comprobando cual puede ejecutar. Lo normal es insertar uno o varios NodesBoolean en el orden que hayan de ejecutarse y poner último uno por defecto.
    - NodeTemporizador: Se inicializa con un tiempo, y sus hijos se ejecutan pasado este tiempo.

- *ComportamientoEnemigoBT*: Tipo de comportamiento de un enemigo que usa un árbol de comportamiento para tomar las decisiones de qué hacer en cada momento.
- *gestorComportamiento*: Gestor que se encarga de crear e inicializar los comportamientos de los enemigos. Un enemigo, cuando despierta, llama al gestor para que le indique qué comportamiento es el que debe seguir.

El enemigo le pide al gestor que le dé un comportamiento en función del tipo que es. Éste le devuelve un ComportamientoEnemigoBT, que contiene internamente el árbol y le va diciendo al enemigo qué debe hacer en las llamadas de la partida *actuar* y *actualizar*.

El árbol de comportamiento se realiza diseñándolo primero en papel, teniendo en cuenta que es lo que queremos que el enemigo realice. Podemos componer los nodos de las mil maneras que queramos, simplemente haciéndolos unos hijos de otros. Por ejemplo, podemos querer tener una secuencia que consista en: primero repetimos disparos al jugador, luego vamos a un punto determinado y realizamos otra secuencia.

La potencia de comportamientos con este sistema es bastante grande, puesto que el comportamiento lo podemos hacer tan grande y tan

complicado como queramos. Solamente tenemos que tener en cuenta que es lo que hace cada nodo.

- **Lógica**

La lógica, propiamente dicha, la componen aquellas entidades que son necesarias para el desarrollo del juego. Las más importantes son:

- *FachadaLogica*: Clase que gestiona los observers de la partida y actúa como interfaz de métodos de la partida.
- *Partida*: Clase que gestiona todos los eventos y todo lo relacionado con la partida. Se encarga de procesar las acciones del jugador y maneja el bucle principal del juego.
- *Avatar*: Clase padre a los personajes del juego. Representa al jugador y a los enemigos.
- *Jugador*: Clase que representa a la entidad lógica del jugador. Actúa según la entrada que da el jugador y los eventos de la partida. Esta clase se encarga también de mandar las señales de visión para la inteligencia artificial.
- *Enemigo*: Clase que representa lógicamente todo lo que tiene que ver con los enemigos. Actúa según el comportamiento le dice que actúe.
- *Asteroide*: Clase que representa un asteroide en el espacio, fragmentándose cuando estalla.
- *Item*: Clase que representa un ítem que puede ser recogido por el jugador, de tipo potencia, salud o una vida.
- *Proyectil*: Representa a los proyectiles de la partida, moviéndose por su trayectoria, y gestionando su eliminación.
- *Explosión*: Clase que representa una explosión cuando muere una nave o una torreta.
- *gestorNiveles*: gestor que se encarga de leer los datos referentes al contenido de un nivel, es decir, toda la información importante relativa a un nivel concreto, como es la malla visual, la posición de

las entidades y donde está el trigger de cambio de nivel, entre otras cosas.
- *staticDataManager*: Manager que se encarga de leer toda la información estática del juego, como son los datos de los enemigos, las opciones iniciales o los tipos de ítems.
- *enemiesGenerator*: Clase que se encarga de generar un número determinado de enemigos cuando se activa.
- *Vector*: Clase que implementa funcionalidades auxiliares para el manejo de vectores en dos y tres dimensiones, como son el producto escalar, la suma de vectores o girar vectores sobre un eje de coordenadas.

6.2 Modelos Gráficos en 3D

Para los modelos en tres dimensiones se ha utilizado el programa privativo 3ds Max, el cual tiene una comunidad de usuarios en internet muy amplia, y además, existen muchos plug-ins que permiten una gran capacidad de personalización del entorno. Cabe decir, además, que tanto en el entorno del desarrollo de videojuegos como en el de la animación, o incluso en la arquitectura, es el programa más usado.

Además de para el modelado, se ha reutilizado un plugin de un proyecto anterior del autor, para editar los escenarios dentro del propio 3ds Max.

El motivo principal de utilizar este software y no otro libre es que ya conocía el funcionamiento del programa y me sentía más cómodo usándolo.

Los modelos se han visto previamente en el capítulo anterior.

6.3 Arte 2D

Para el arte 2D se ha utilizado indistintamente Gimp y Photoshop. El primero es un programa de software libre que es ampliamente usado en todo tipo de proyectos. Tiene una gran capacidad de edición, y está muy optimizado. El segundo, al tener una gran tradición detrás, ofrece unos resultados algo más profesionales que el primero, como es en el resultado de aplicar una serie de filtros.

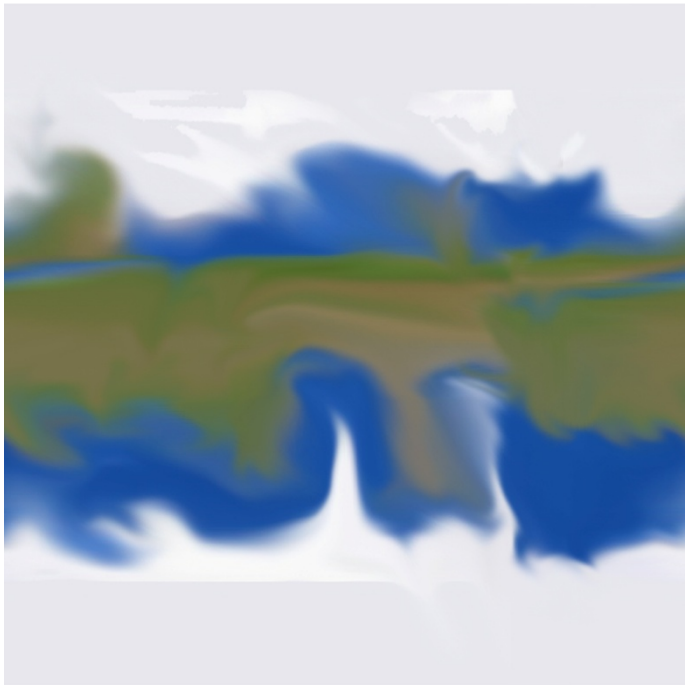

Figura 6.4. Ejemplo de textura 2D.

Como ejemplo tenemos la imagen 6.4 que muestra un ejemplo de textura en 2D. En este caso es la textura del planeta que se ve de fondo en la fase 1. Esta textura se aplica como si fuera papel pintado sobre el modelo tridimensional para darle color.

Destacar que se ha usado el segundo más por conocimiento del programa que por demérito de Gimp, que lo conozco menos.

6.4 Sonido

Para el sonido se ha escogido el motor Fmod, puesto que ya había trabajado en otros proyectos y tengo cierta experiencia. Así pues, teniendo la posibilidad de utilizar cualquier otro motor, como la propia SDL que se usa para la entrada del joystick, se ha preferido utilizar el motor conocido.

Todos los sonidos utilizados están bajo las licencias de Creative Commons, que son unas licencias que están inspiradas en la licencia GPL (General Public License) de la Free Software Foundation. El objetivo de estas licencias es facilitar la distribución y el uso de cualquier tipo de contenido, incluido el multimedia.

Capítulo 7

Pruebas

En todo desarrollo de videojuego las pruebas lo son todo. Los equipos de QA (Quality Assurance) prueban que está todo correcto desde las fases más tempranas del juego. Existen diferencias sutiles entre una fase de pruebas de un desarrollo de videojuegos y un software de gestión al uso.

En un programa de gestión se realizan primero pruebas sobre los casos de uso. Se comprueba que se cumplen cada uno de ellos, para luego pasar a realizar pruebas de caja negra, mirando los valores de entrada y de salida, y las de caja blanca, mirando el flujo de la aplicación. Por último, se deja al usuario que pruebe para ver si realmente se cumplen las especificaciones pactadas.

En un desarrollo de videojuegos, la principal diferencia es que puede o no haber usuario. Lo normal es que no haya, sino que son los diseñadores los que piden las especificaciones. Y el proceso de pruebas es diferente, como se describe a continuación, por orden de modularidad, realizado sobre el proyecto.

7.1 Pruebas de Diseño

Al diseño se le han hecho pruebas de lectura, comprobando que cada aspecto resultaba lo más claro posible a la hora de indicar los siguientes aspectos:

- Claridad

Un aspecto que hay que comprobar en un buen diseño es que sea lo más claro posible, no presente ninguna ambigüedad y, a ser posible, lo más conciso que se pueda, pero sin omitir ningún detalle relevante. Que sea lo más claro posible favorece la comunicación entre diseñadores y el resto del equipo a la hora de trazar un buen plan de desarrollo, y sobre todo, un plan de pruebas.

- Viabilidad

La viabilidad de un documento de diseño viene determinada por dos factores principales: el tiempo y los recursos humanos, entendiendo recursos humanos como personal y dinero. La manera más óptima de probar que el diseño es viable es realizar una estimación de tiempo en relación a todas las funcionalidades. Una buena planificación determina si un proyecto es viable o no.

- Consistencia

Un documento ha de ser consistente. Esto es lo más difícil de probar, ya que es difícil de detectar. Un documento inconsistente es la mayor fuente de errores posibles en cuanto a diseño, ya que por muchas veces que se lea el documento, si te quedas con una idea de la parte inconsistente, se puede estropear el resto del desarrollo. Así que un gran porcentaje de esfuerzo de las pruebas sobre diseño debería ir en esta dirección.

En este caso, como diseñador, programador y QA ha sido la misma persona, no ha habido problemas de documentación en este sentido.

7.2 Pruebas de Prototipado

Durante la fase de prototipado, se realizan muchos prototipos, cada uno de los cuales implementa una o varias funcionalidades, las cuales han de ser sometidas a pruebas intensivas e incrementales.

A medida que se van probando, se van introduciendo nuevas y se prueban éstas junto a las anteriores para ver si no hay inconsistencia entre ambas. Podemos afirmar que es un desarrollo de tipo iterativo-incremental, por lo que a cada funcionalidad nueva se le debe realizar un proceso nuevo.

Así se ha hecho durante el desarrollo. Primero se construyó la arquitectura básica, y se probó que el bucle principal de juego se ejecutaba. Después se fueron introduciendo personajes y probando sus mecánicas. Luego la interacción con ítems, y así hasta la última funcionalidad probando cada una de ellas exhaustivamente.

7.3 Pruebas de Codificación

Una vez todas las funcionalidades han sido probadas lo suficiente, se pasa al siguiente nivel que es terminar el código. En ese punto, hay que probar que las mismas mecánicas no dejan de funcionar al introducir todos los cambios que quedan pendientes. Estas pruebas se realizan con gráficos lo más feos posibles de manera que no se puedan confundir con los gráficos finales. Durante este paso es cuando más errores se detectan puesto que interactúan las funcionalidades ya creadas con el resto del código.

Los errores más comunes suelen ser comportamientos extraños o alteraciones de las rutas de los personajes.

7.4 Pruebas de Juego

Una vez que todas las mecánicas se han probado lo suficiente y se han dado por válidas, llega el momento de meter el contenido en el juego. Se terminan de introducir todos los modelos gráficos, se observa como encajan y se utilizan las herramientas auxiliares para darle vida al juego. El objetivo de las pruebas aquí es concretar que en el juego, la estética visual es la correcta y que todos los modelos encajan correctamente.

Los mayores problemas que se encuentran aquí suelen ser modelos gráficos mal exportados, desbordamientos de memoria o tasa de frames por debajo de lo normal.

7.5 Pruebas Finales

Las pruebas finales se realizan una vez tenemos la versión Gold, para asegurarnos de que se instala correctamente, el juego final se ejecuta en correctas condiciones y sobre todo que el producto que se entrega es correcto. Si pasa la prueba se convierte en Gold Master y es la versión final que se entrega al usuario. Esto no significa que el resultado final esté exento de errores, sino que los que existen no son bloqueantes ni graves.

Si se detecta que una vez pasado a Gold Master la versión encuentra algún fallo, se crean parches para solucionar los problemas. El desarrollo tiene que prever la existencia de parches y estar preparado, por si luego hay que aplicar alguno.

Capítulo 8

Glosario

3ds Max. Programa para Mac y Windows de modelado y animación de gráficos tridimensionales, muy utilizado en películas de animación y videojuegos.

DDS. De las siglas DirectDraw Surface es un estándar de compresión de imágenes, especialmente compatible y optimizado para DirectX.

DirectX. Colección de interfaces de aplicación desarrolladas para facilitar las complejas tareas relacionadas con multimedia, especialmente programación de juegos y vídeo, en la plataforma Microsoft Windows.

Frame. Del inglés marco, fotograma, es una imagen concreta dentro de una sucesión de imágenes que forman una animación. La suma de estas imágenes es lo que produce la sensación de movimiento. La velocidad de una animación se medie en frames por segundo (fps).

GIMP. GNU Image Manipulation Program, es un programa de edición de imágenes digitales en forma de mapa de bits. Es un programa libre y gratuito, forma parte del proyecto GNU y está disponible bajo la Licencia pública general de GNU.

JPEG. Del inglés *Joint Photographic Experts Group*, es un método de compresión de imágenes, que utiliza un algoritmo de compresión con pérdida para poder reducir el tamaño de las imágenes.

Librería. Conjunto de programas externos a la aplicación que se esté desarrollando. Suelen ser ficheros que contienen código que se añade a la aplicación para que ésta lo ejecute. Normalmente se añaden en el proceso de vinculación, aunque algunas veces se incluye el código a la compilación.

Malla. Representación virtual de un objeto en tres dimensiones. Una malla está formada por vértices que se conectan entre sí mediante aristas, formando un conjunto de planos que representan un volumen.

OGRE. Motor de renderizado en tres dimensiones, multiplataforma, de libre distribución y escrito en C++. Se ha utilizado en varios juegos comerciales, como por ejemplo Torchlight.

Photoshop. Programa de edición de imágenes digitales privativo.

Place holder. Gráfico intermedio que se usa en el desarrollo de un videojuego para indicar que es un gráfico que ha de ser sustituido por el final. Lo ideal es que sea un gráfico lo peor hecho posible o que no pegue nada con el entorno, para que sea claramente identificable.

Quality Assurance (QA). Departamento de un grupo de desarrollo que se encarga de probar la aplicación y verificar que no tiene errores y el programa funciona como debe.

Script. Del inglés guión, es una serie de instrucciones que un programa lee y se encarga de ejecutarlas una a una. En videojuegos se suele utilizar para definir inteligencias artificiales.

SDL. Del inglés, Simple DirectMedia Layer, es un conjunto de librerías que proporcionan una serie de funciones que se encargan del manejo y carga de imágenes, sonidos, música y entrada de joysticks.

Sprite. Dibujo que forma parte de una serie que, mediante la reproducción de cada uno de ellos en un orden específico, se consigue una sensación de animación. Este recurso se utiliza en videojuegos en 2 dimensiones al ser imágenes planas.

Textura. Imagen que se aplica sobre una estructura en dos dimensiones o tres, para darle visionado.

Capítulo 9

Conclusiones

A continuación hablaremos globalmente del proyecto, que puntos fuertes tiene, según el punto de vista del autor, hablaremos de la experiencia personal que supone hacer un proyecto de este calibre y añadiremos algunas mejoras que podrían ser implementadas.

9.1 Análisis

Lo primero que destacaría de todo el proyecto es la extensísima duración que ha tenido. La planificación que se ha incluido es de este proyecto, pero hay que decir que este proyecto viene de antes. Podría decirse que he hecho tres proyectos en vez de uno. Es decir, comenzó siendo un juego de PC para Linux, en c++ con SDL. Por circunstancias de la vida, dejé de lado ese proyecto, para empezar uno nuevo en c++ con Nebula2 para Windows. Y por retrasos varios, dificultades inesperadas y otros motivos volví a tirar el proyecto una vez más. Esta vez sí, paso a ser c++ con OGRE en Windows. Aunque de todas maneras, pese a estar la plataforma decidida y el lenguaje también, tuve que volver a tirar de nuevo el proyecto y "comenzar" de nuevo. Lo cierto es que no tuve que recomenzar de cero, puesto que este último cambio de rumbo no fue tan grande como los dos anteriores. La planificación comienza en este punto.

En cuanto a funcionalidades qué destacaría, sin duda, la que más me enorgullece, es el sistema de inteligencia artificial que he implantado. Por la estructura del proyecto, lo cierto es que no luce mucho, pero tiene una potencia considerable, demostrado en otro proyecto. La estructura de Árboles de comportamiento proporciona un interfaz claro, muy fácil de expandir y muy fácil de crear comportamientos.

Otra cosa a destacar es el modo Arcade. Un modo de juego que permite alargar la vida del juego más allá de la historia, puesto que el jugador puede jugar todo el rato que quiera hasta que se canse o se le terminen las vidas.

Sobre las herramientas que destacan, la que más es un visor debugger de PhysX al que puedes conectar tu aplicación, para que te muestre que es lo que está pasando en el mundo físico. Esto, sin duda, es muy valioso, porque ajustar la malla física a la malla visual, es una tarea muy costosa. Y sobre todo sirve para ver qué pasa con las entidades, ya que es muy común que se queden entidades "fantasma" y estorben en la normal ejecución del juego.

9.2 Posibles mejoras

La primera mejora que necesita este juego son más recursos gráficos. La penalización de no ser grafista y no poder conseguir los gráficos necesarios, ha mermado muchísimo las posibilidades del juego. Aún así, ha quedado un resultado razonablemente aceptable. Lo importante sería contar con modelos más detallados y, sobre todo, más recursos de escenarios, puesto que únicamente se han podido realizar dos fases.

Otra mejora podría ser, quizás, la implementación de alguna herramienta para el diseño de los personajes. Los datos estáticos han sido diseñados con el bloc de notas, probando valores. Por falta de tiempo no se ha realizado una herramienta, que por una parte no

hubiera sido difícil de implementar, y por otra pudiera ser de mucha utilidad.

Una gran mejora en este juego supondría, quizás, la inclusión de multi-jugador online. Junto al modo arcade serviría para darle mucha vida al juego, ya que es divertido matar naves, pero lo es aún más con alguien humano. Y si es una lucha entre naves, más divertido es aún.

9.3 Experiencia personal

La experiencia personal que puedo sacar de este proyecto es mucha, ya que normalmente un videojuego supone un gran esfuerzo de sacar adelante y, sobre todo, de cerrar. Saber cerrar oportunamente cualquier videojuego es lo más difícil que uno se puede encontrar en esta profesión. Afortunadamente he tenido la oportunidad, profesionalmente y académicamente de poder cerrar más de un videojuego, lo cual supone para mí un plus de satisfacción, sabiendo que he podido terminar uno más.

Cuando un proyecto se acaba, te invade una sensación de vacío al principio, ya que parece que toda esa gran cantidad de tiempo, que al final es cantidad de vida, que se ha dedicado a crear un sistema, que ahora tiene que andar sólo, parece que ha sido en vano. Pero nada más lejos de la realidad, porque aunque el juego sea malo, o no sea lo suficientemente bueno, siempre se aprende. Se pueden aprender cosas tangibles (un lenguaje determinado, un programa, a veces incluso un kit de desarrollo) o bien, lo que para mí tiene más valor, cosas intangibles. Cosas que no se pueden cuantificar, pero que están ahí. Poder trabajar en un estudio profesional, codo con codo con gente que tiene una gran trayectoria detrás, a la cual tú esperas poder llegar algún día tener; saber que esa gente tan humilde ha hecho juegos con los que tú has crecido jugando, te tratan de igual a igual, eso no tiene precio. Ese valor humano, quizás, es lo mejor de un desarrollo de videojuegos.

Poder cerrar un videojuego, sabiendo que las cosas están bien hechas, es la esperanza que todo desarrollador de videojuegos tiene en su corazón para seguir trabajando en este mundo día a día. El juego luego puede ser bueno o malo, pero el trabajo queda, y la experiencia de haber trabajado, con quién se ha trabajado, es muchísima. Las ventas o los premios luego tienen su valor, pero desde luego apenas se puede comparar.

Apéndices

A. Documento de concepto

1. Ficha técnica

Plataforma: PC

Género: Acción/ Shoot' em up/ Naves

Edad: +13

Jugadores: 1

2. Descripción

Juego de naves en tercera persona sobre raíles. El jugador debe manejar su nave espacial y matar cuantas más naves mejor. El objetivo del juego es llegar hasta el final de los niveles consiguiendo el mayor número de puntos. Dispondrá, además, de un modo "Arcade" en el que se enfrentará a muchas naves espaciales sin parar, hasta que muera o termine la partida.

3. Ambientación

Estamos en el año 2457 d.C. Las tropas intergalácticas estelares dominan el comercio de toda la galaxia. El ejército Xel' thalá de la

galaxia UB543b ha conseguido llegar a esta galaxia en pos de conseguir el objetivo de dominar el universo. Con lo que no contaban es que su mayor guerrero estaba fuera en misión de reconocimiento. Cuando vuelve de su misión encuentra que las tropas enemigas están arrasando su planeta y no lo puede permitir…

4. Mecánicas centrales

- Movimiento

La nave del jugador se mueve en línea recta por el eje de coordenadas Z a una velocidad constante. El jugador puede moverse con una limitación en altura y anchura libremente.

- Disparo

El jugador tiene una única arma que es el disparo de proyectiles. Se lanza de dos en dos y tiene diferente potencia, según los ítems que haya ido recogiendo.

- Aceleración y freno

El jugador puede acelerar la nave y frenarla a su antojo apretando el botón correspondiente. La aceleración y freno es sutil con respecto a su velocidad constante.

- Recoger ítem

El jugador puede recoger una serie de ítems a lo largo del escenario. Cada ítem le otorga un beneficio distinto, cuyos efectos son: recuperar energía, aumentar la potencia, aumentar los puntos de la partida o incrementar el número de vidas.

5. Referentes

- Starfox64 (Nintendo, 1997). Juego de naves en tercera persona en el que encarnamos a un mercenario espacial, al que le encargan

diversas misiones para salvar diversos planetas. Según el jugador va pasando niveles, y en función de lo bien que lo haga, podrá ir por un camino u otro.

- Stardium Space arena (Máster en desarrollo de videojuegos, UCM, 2010). Juego de naves en tercera persona, multijugador. Estamos en un programa de televisión espacial, donde varias naves se enfrentan para ganar el concurso. Aquí el objetivo, más que matar cuantas más naves mejor, es aguantar vivo hasta el final.

- Level1 (Grupo2, Máster en desarrollo de Videojuegos, UCM, 2010). Juego de plataformas y acción en el que somos 4 hermanas que utilizan el poder de los elementos para crear diversas armas. Escoge bien que elemento quieres usar, porque cada situación requerirá una estrategia diferente. Cuenta con un modo "Misiones", de estilo VR-missions del Metal Gear Solid. La estética de éste modo es lo que queremos poner para el modo arcade.

6. Riesgos
    - Los gráficos. Por falta de artistas, los gráficos es el punto débil de este proyecto, ya que aunque no hace falta una gran cantidad de recursos gráficos, sí que son necesarios algunos y que estén hechos relativamente bien. Se intentará buscar recursos externos.
    - El tiempo. Dada la magnitud del juego, al estar disponible un solo programador para realizarlo, pudiendo alargarse demasiado el desarrollo en el tiempo. La solución a este problema pasa por intentar invertir todo el tiempo disponible en el desarrollo.

B. Manual de instalación

Para poder instalar el juego, hay que ejecutar el archivo instalador, llamado StarOgre.exe. Si hacemos doble click sobre el instalador se nos abrirá y aparecerá la ventana de bienvenida.

Figura B.1. Pantalla de bienvenida a la instalación.

Haciendo click en Next, nos lleva a la pantalla donde se muestra la licencia del programa y se nos pregunta si estamos de acuerdo con la licencia. Haciendo click en “I agree”, “estoy de acuerdo”,

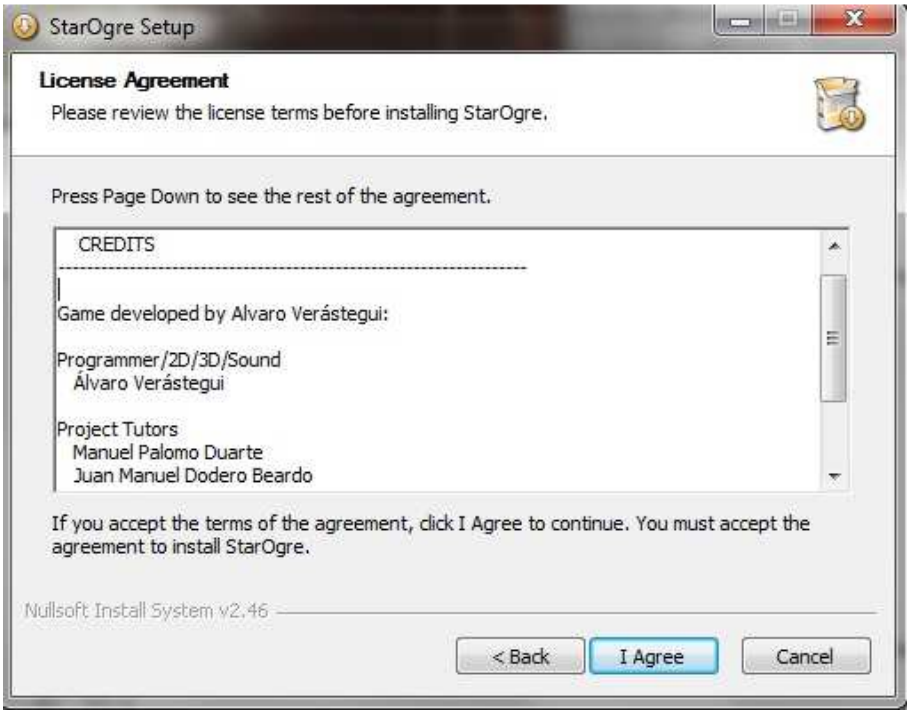

Figura B.2. Pantalla de la licencia de la instalación.

nos lleva a una pantalla donde se nos muestra información acerca de cuanto espacio se requiere, y cuanto nos queda disponible, además de que ruta queremos poner para la instalación.

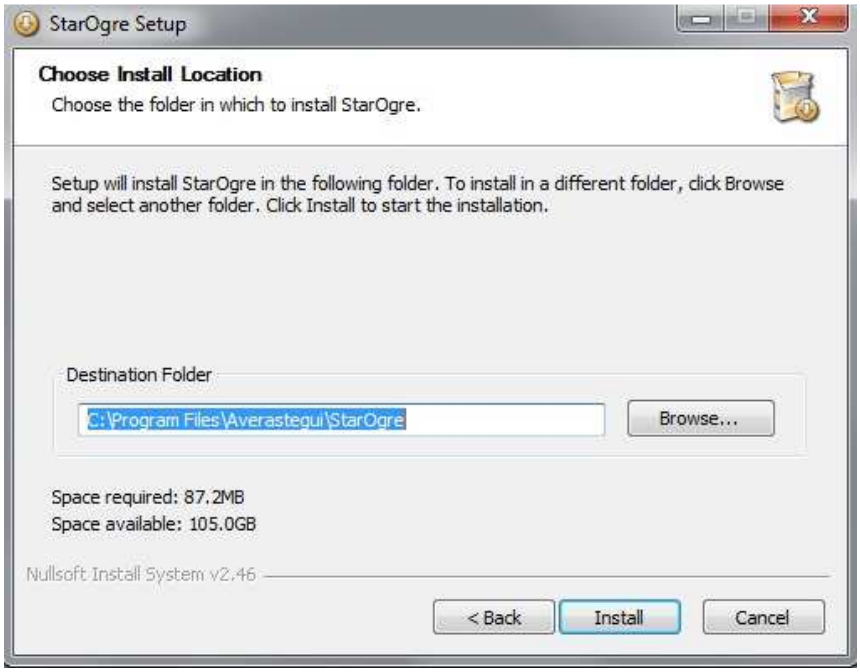

Figura B.3. Pantalla de espacio disponible y ruta de la instalación.

Si hacemos click en instalar, nos lleva a la pantalla de información de la instalación en curso. Esta ventana instala las dependencias del juego, que son algunas librerías de visual studio, PhysX y la versión 9 de DirectX. Si ya tenemos instalada alguna versión posterior a las que se van a instalar se omite ese paso.

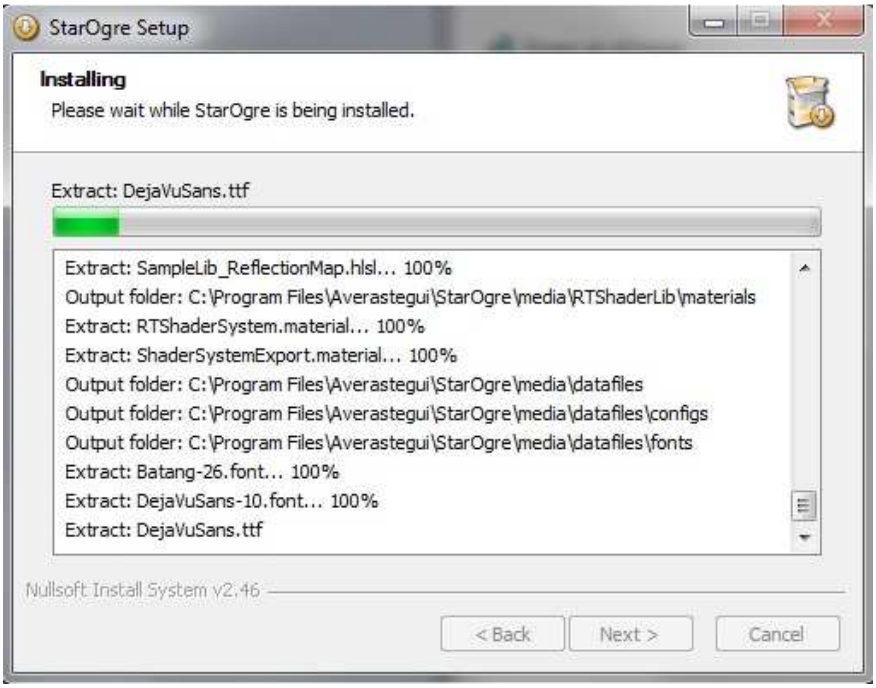

Figura B.4. Progreso de la instalación

Cuando termine la instalación, si todo ha salido correctamente, aparecerá la ventana de confirmación y una vez dándole a finalizar ya tenemos instalado el juego y sólo nos queda ejecutarlo para empezar a jugar.

Figura B.5. Pantalla de confirmación de la instalación.

Para desinstalar el juego basta con ejecutar el desinstalador que se encuentra en el menú inicio, o en la carpeta del juego. Si lo ejecutamos, aparecerá una ventana que nos pedirá confirmación de que queremos realmente desinstalar el juego.

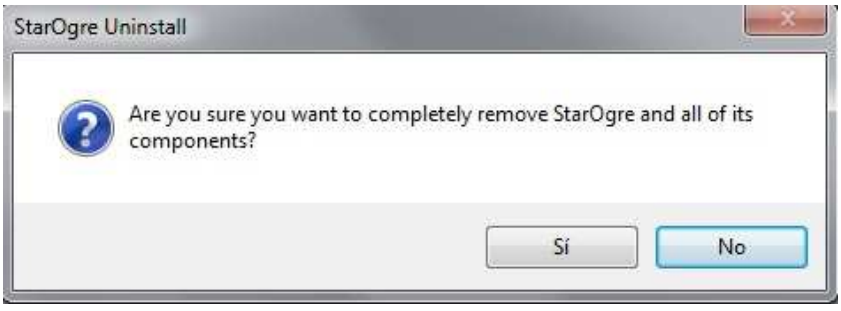

Figura B.6. Pantalla de confirmación de desinstalación.

Si decimos que sí, el juego se comenzará a desinstalar mientras nos muestra la pantalla de progreso.

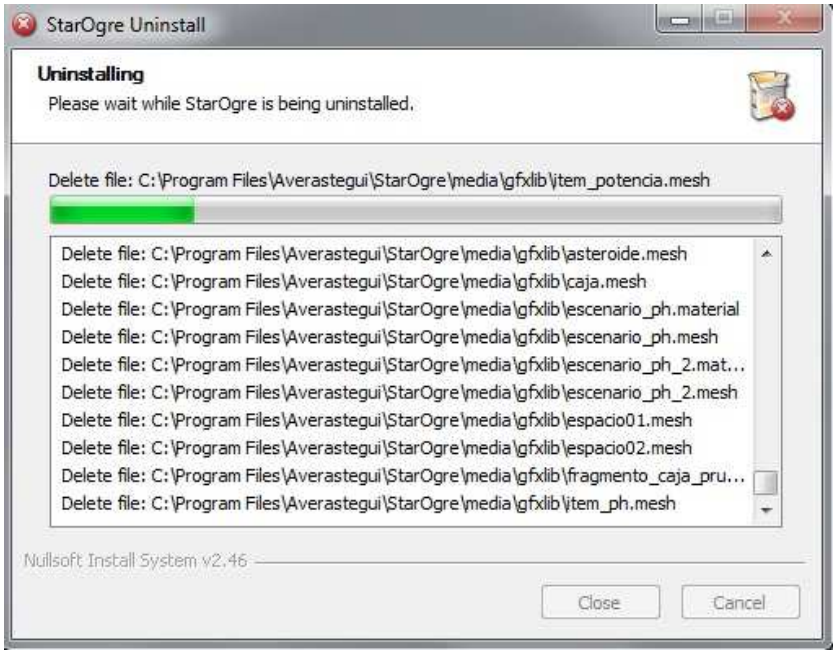

Figura B.7. Pantalla de progreso de la desinstalación.

Cuando termina la desinstalación el botón de cerrar se desbloquea y pulsándolo cerramos la ventana finalizando la desinstalación.

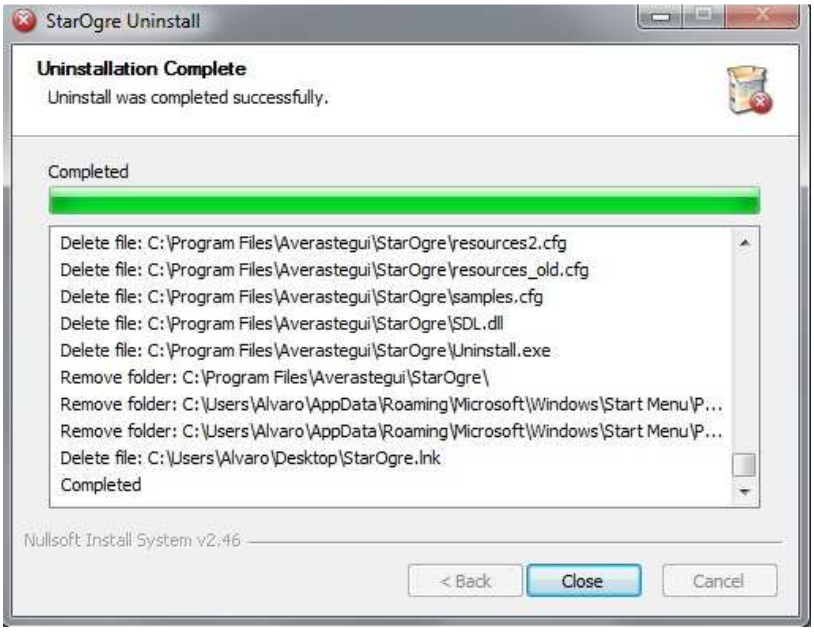

Figura B.8. Finalización de la desinstalación.

Y con eso estaría desinstalado de nuestro ordenador.

C. Manual de Usuario

1. Controles

Para jugar se puede escoger entre el teclado o un joystick para jugar. En función de que opción escoja los controles son los siguientes:

- Teclado

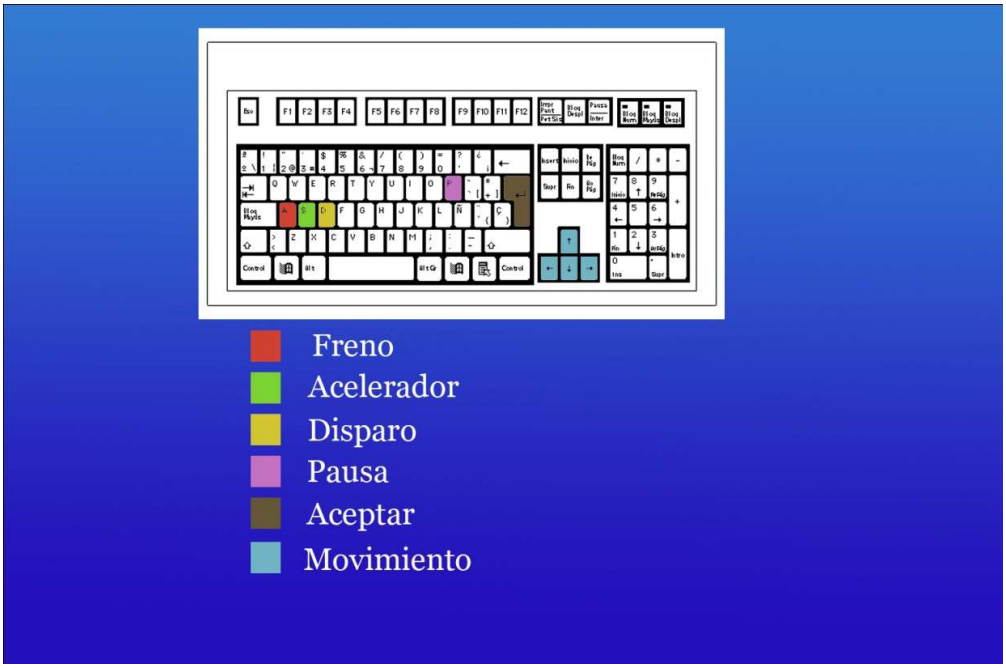

Figura C.1. Controles del teclado.

- Joystick

Figura C.2. Controles del Joystick

2. Opciones

Antes de empezar a jugar, podremos escoger las opciones principales, sonido, controles y resolución, ver los récords del juego o ver los créditos. Para cambiar las opciones principales tenemos que seleccionar la opción “opciones” en el menú principal, este nos lleva a la pantalla de menú de opciones. Seleccionamos la que queramos cambiar y escogemos su valor.

Figura C.3. Menú principal

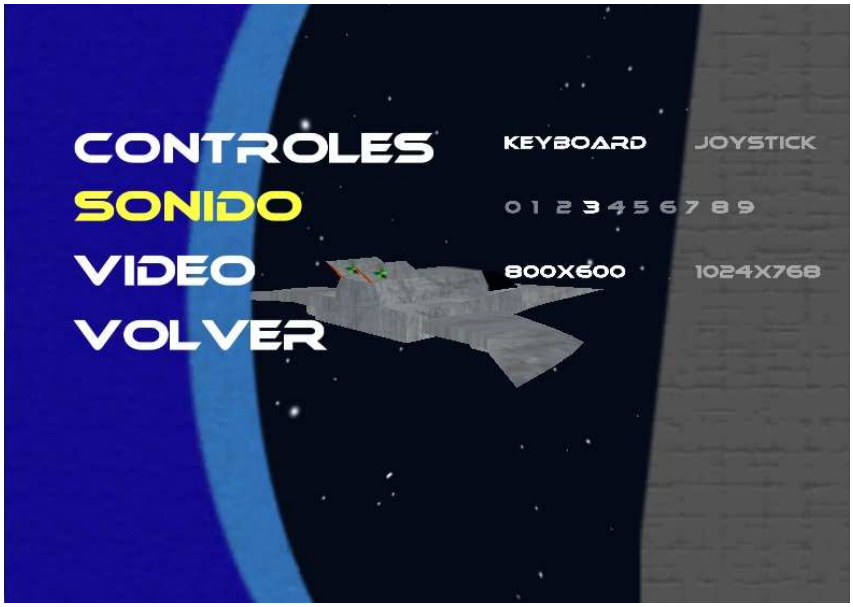

Figura C.4. Menú de opciones

Si lo que queremos ver son los créditos o los récords del juego, en el menú principal tenemos que escoger la opción que queramos.

Figura C.5. Créditos del juego

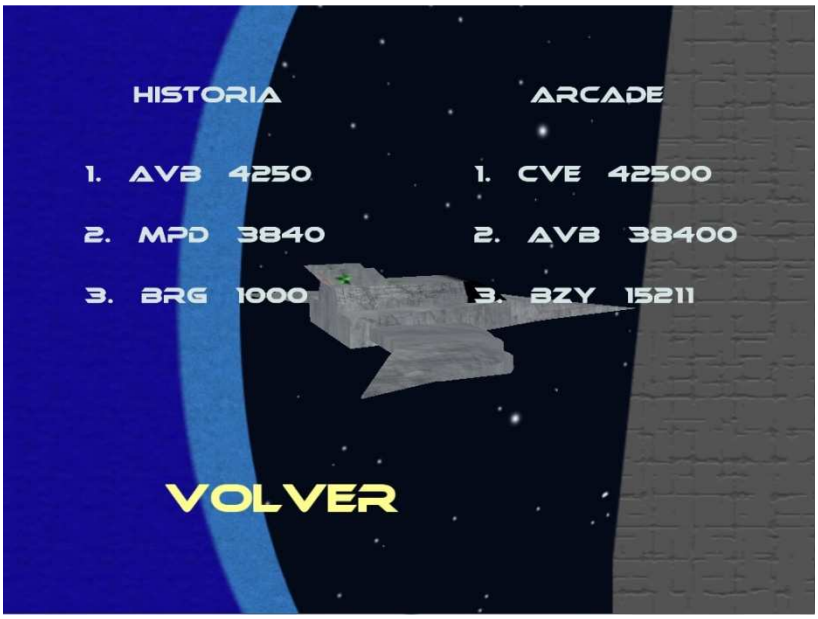

Figura C.6. Récords

3. Juego

Tenemos dos modos principales de juego, el modo Historia en la que hay que completar dos escenarios, intentando llegar al final intentando conseguir la mayor puntuación posible o el modo Arcade, que consiste

en un escenario infinito en el que no paran de aparecer naves hasta que se pierde la última vida. Para escoger el modo de juego, hay que seleccionar la opción jugar del menú principal, y luego en el menú de juego una de las dos opciones posibles, o bien, seleccionar la opción volver para cambiar alguna opción en el menú de opciones.

Figura C.7. Menú de Juego.

Durante la partida, podemos pausarla por si queremos descansar, especialmente recomendable en el modo Arcade, simplemente pulsando el botón de pausa. Desde este menú podemos volver a la partida o bien, salir del juego pero, ¿Por qué querrías dejar de jugar?

Figura C.8. Menú de pausa.

Durante el juego tenemos en la pantalla unos elementos informativos, llamados HUD, del inglés Head Up Display, que nos dan la información necesaria de la partida.

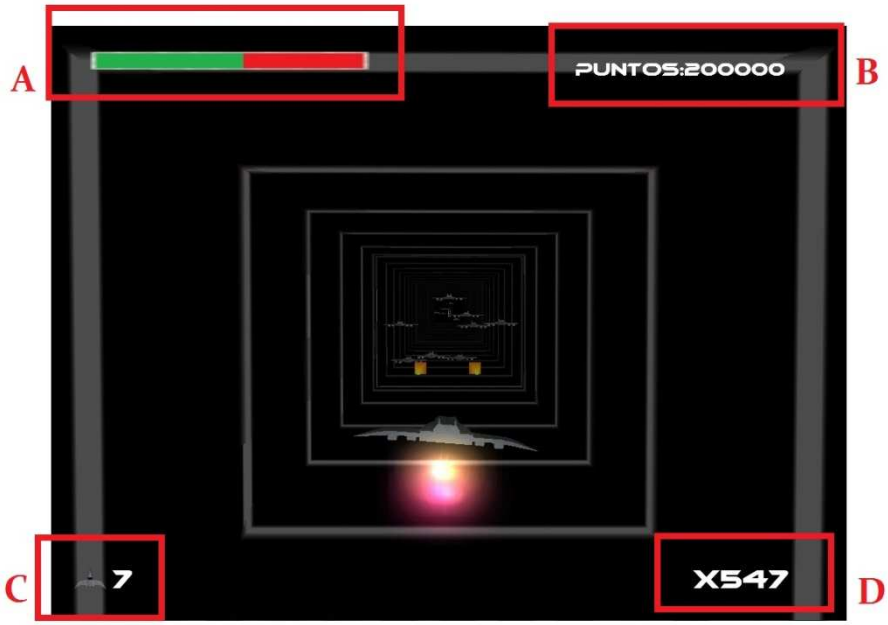

Figura C.9. HUD del juego.

Como se aprecia en la imagen, hay cuatro elementos importantes, uno en cada esquina de la pantalla, marcados con las letras A, B, C y D. Estos elementos son:

- A: Barra de vida, nos indica la vida restante de la nave. La barra verde es la energía restante y la roja la energía que hemos perdido.
- B: Puntos de la partida.
- C: Número de vidas restante.
- D: Número de naves eliminado.

Para eliminar las naves enemigas, el jugador tiene un arma que dispara proyectiles. Para ayudarle en la tarea, existen unos ítems que el jugador puede recoger para otorgarle ciertos beneficios. Existen cuatro tipos:

- Potencia: Mejora el arma.
- Salud: Recupera energía.
- Puntos: Incrementa la puntuación de la partida.
- Vida: Suma una vida al contador de vidas.

Nos podemos encontrar dos tipos de enemigos durante la partida, enemigos terrestres y enemigos aéreos. Del buen pilotaje del jugador depende la victoria frente al ejército enemigo. Buena suerte, será necesaria…

4. Historia

Estamos en el año 2457 d.C. Las tropas intergalácticas estelares dominan el comercio de toda la galaxia. El ejército Xel' thalá de la galaxia UB543b ha conseguido llegar a esta galaxia en pos de conseguir el objetivo de dominar el universo. Con lo que no contaban es que su mayor guerrero estaba fuera en misión de reconocimiento. Cuando vuelve de su misión encuentra que las tropas enemigas están arrasando su planeta y no lo puede permitir…

¡Deprisa StarOgre, el destino de tu planeta depende de ti!

D. Post-Mortem

1. Background
   - El juego forma parte del proyecto de fin de carrera del alumno de la universidad de Cádiz, Álvaro Verástegui, para terminar la carrera de Ingeniería Técnica en Informática de Sistemas.
   - El desarrollo del juego está hecho por completo por el alumno, excepto los recursos de Sonido. Tanto arte 2D/ 3D cómo el código del juego está programado enteramente por él, a excepción de las librerías externas.
   - El juego toma lugar en el espacio donde controlamos a una nave espacial en busca de venganza por haber invadido el planeta natal del piloto.

2. Ficha técnica

Desarrollador: Álvaro Verástegui

Tutor del proyecto: Manuel Palomo

Tiempo de desarrollo: 5 meses.

Fecha de release: 14 de julio de 2011.

Tecnología: OGRE 3D, PhysX, Fmod.

Hardware: Intel Core 2 Duo, 2 GB Ram, 100 Mb Disco Duro.

3. Qué fue bien
   - Experiencia previa. Una de las grandes ayudas para conseguir hacer este proyecto ha sido sin duda la experiencia previa de haber realizado el máster en videojuegos de la universidad complutense de Madrid. Durante el máster se realizó un

videojuego, que aunque fue con otro motor de renderizado, el cómo crear una arquitectura y las bases del juego sí que se pudieron mantener.

- Editor escenarios. Durante el proyecto anterior, se realizó por parte del grupo un editor de escenarios consistente en un plug-in de 3ds Max, que permitió editar el escenario directamente sobre la visualización de éste. Este plug-in se ha reutilizado para este proyecto, modificando sus parámetros.
- Comunidad de OGRE. OGRE posee una enorme comunidad abierta en internet que comenta las soluciones a problemas que se van encontrando durante el desarrollo de sus aplicaciones. Esto ha sido de gran ayuda para problemas puntuales.

4. Que fue mal
    - Falta de grafismo. Al no disponer de un artista, el único programador que se encuentra desarrollando ha tenido que hacer las veces de grafista, lo cual resta tiempo en aprendizaje y desarrollo de los gráficos, que se podía utilizar en programación. Esto ha supuesto, sin ninguna duda, el gran lastre al proyecto.
    - Cambios. Al no disponer de grafista, el proyecto ha sido cancelado y retomado de cero en varias ocasiones, cambiando radicalmente de proyecto. Por haber estado rediseñando el juego en tantas ocasiones, el diseño del juego final puede no ser tan bueno como cabría esperar.
    - Tiempo. Además de no tener grafismo, al programador lo contrataron en Pyro Studios, mermando la cantidad de tiempo que podía dedicar al desarrollo, por lo que el proyecto se ha extendido en el tiempo más de lo deseado. Esto, junto a la poca habilidad con las artes gráficas han sido los grandes problemas del desarrollo.

Bibliografía

[Ogre3D] The OGRE Team. Ogre3D – Open source 3D Graphics Engine. http: //www.ogre3d.org Julio 2010 – Julio 2011

[OgreWiki] Ogre3D community. Ogre Wiki. http://www.ogre3d.org/tikiwiki/ Julio 2010 – Julio 2011

[OgreTut] Ogre 3D Tutorials. http://www.ogre3d.org/tikiwiki/Tutorials Julio 2010 – Julio 2011

[MdvUcm] Apuntes del máster. Máster en desarrollo de Videojuegos. Escuela Politécnica Superior. Universidad Complutense de Madrid

[AIArtifIn] Artificial Intelligence for Games. Ian Millington. ISBN-13: 978-0-12-497782-2

[AIWisdom] AI Game Programming Wisdom. Steve Rabin. ISBN: 1-58450-077-8

[FundCpp] Fundamentos de C++. F. Palomo Lozano, I. Medina Bulo, G. Aburruzaga García. ISBN-13: 978-8498280074

[WikiEs] Wikipedia. http://es.wikipedia.org Julio 2010 – Julio 2011.

Licencia

GNU Free Documentation License
Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsable for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as

a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you
copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input
to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or non commercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of

these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section

of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section
may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements"
or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant

Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the

compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a
translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the ocument specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published not as a draft) by the Free Software Foundation. If the Document does not specify a versión number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation;

with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with…Texts." line with this:

with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.